ASHTON·TATE

# USING dBASE III PLUS

# dBASE III PLUS

Copyright © 1985, 1986 Ashton-Tate
As an unpublished work. All Rights Reserved.

Use of the software contained in this package has been provided under a Software License Agreement. Please read it thoroughly. In summary, you may not make copies of the software except as specifically permitted under the Software License Agreement. You may use the software only on a single terminal or a single workstation of a computer (or its replacement): accordingly, you must license a separate copy for each terminal or workstation where you want to use the software.

Note: Unauthorized use of the software or of the related materials can result in civil damages and criminal pernalties.

The software and related materials in this package are licensed with a Limited 90-Day Warranty. **Other than the limited warranties that are expressly stated therein, Ashton-Tate makes no other warranty, express or implied, to you or any other person or entity. We will not be liable for incidental, consequential or other similar damages. In no event will our liability for any damages ever exceed the price paid for the license to use the software, regardless of any form of the claim. You may have other rights which vary from state to state.**

Trademarks and Service Marks/Owners: pfs/Software Publishing Corporation; VisiCalc, Lotus, 1-2-3/Lotus Development Corporation; WordStar/MicroPro International Corporation; MS-DOS, Microsoft, Multiplan/Microsoft Corporation; MultiMate/Multimate International Corporation, an Ashton-Tate Company; IBM,
PC-DOS, PC/XT, AT/International Business Machines Corporation; Compaq/Compaq Computer Corporation; Tandy/Tandy Corporation; Intel/Intel Corporation; Leading Edge/Leading Edge Products, Inc; TeleVideo/TeleVideo Systems, Inc. Other product names mentioned in this package may also be trademarks used here for identification purposes only.

dBASE III PLUS and dBASE III PLUS LAN Pack are trademarks, dBASE, dBASE III, and Ashton-Tate are registered trademarks of Ashton-Tate.

6/86

## Index

This is a convenient cross-reference to all the concepts, commands, and functions discussed in your dBASE III PLUS manuals. The index references all the manuals in both Volumes One and Two. A duplicate index is also in Volume Two.

# Introduction

*Using dBASE III PLUS* is an encyclopedia of information on keywords, commands and functions, and how to do basic data management tasks in dBASE III PLUS. If you have completed *Learning dBASE III PLUS* or are otherwise familiar with dBASE, programming, or database management, this section is for you.

If you are an intermediate-to-advanced user, you may also wish to refer to *Programming with dBASE III PLUS*.

This section is organized into seven chapters:

**Chapter 1: Overview**

This contains the technical specifications of dBASE III PLUS and discussions of the basics: file types, field types, memory variables, and the like.

**Chapter 2: Using Commands**

This chapter provides another path for learning the program. It gives you more detail on the information you learned in Chapter 9, "dBASE III PLUS Commands," in *Learning dBASE III PLUS*. You'll find out more about command syntax, operators, handling multiple databases via work areas, relations, or catalogs, and how to organize and search your data to get the facts and results you need. The chapter is divided into task-oriented sections which describe the commands you need to complete basic operations in dBASE III PLUS.

**Chapter 3: About Program Files**

This chapter takes up where Chapter 10 in *Learning dBASE III PLUS* left off. You learn about running dBASE III PLUS by means of program files. With program files you gain more control over your data than you have by using commands only at the dot prompt. This chapter tells you about programming commands and how to create, execute, close, and debug program files.

**Chapter 4: dBASE III PLUS Configuration**

This chapter describes two configuration files that you need to run dBASE III PLUS properly: Config.sys and Config.db. You'll also learn about the SET menu and the different options you have to customize dBASE III PLUS or the programs you create.

**Chapter 5: Commands**

This chapter is the bulk of the manual, as commands are the bulk of dBASE III PLUS. Chapter 5 along with Chapter 6, "Functions," make up the reference section of your manuals. Chapter 5 has a Classes of Commands section in which the commands are divided according to how they're used. The commands are then listed in alphabetical order and thoroughly described. The description includes a definition of the command, its syntax, usage, examples, and any tips that may make it easier for you to use. The files used in the examples are on the Sample Programs and Utilities disk.

**Chapter 6: Functions**

This chapter along with Chapter 5, "Commands," make up the reference section of your manual. Chapter 6 contains the same type of information for each function as Chapter 5. The functions are divided into classes according to how they're used. Each function is also listed in alphabetical order and thoroughly described. The description includes a definition, syntax, usage, examples, and any tips that make it easier for you to use. The files used in the examples are on the Sample Programs and Utilities disk.

**Chapter 7: Error Messages**

Each error message that can appear in dBASE III PLUS is listed here, along with an explanation of possible causes of the message and applicable code numbers.

**Appendices**

These contain:

- A table of control characters and their meanings
- A table of end-of-file and beginning-of-file conditions for indexed and nonindexed files
- The structure of a database (.dbf) file
- A table of printable ASCII character codes

**Glossary**

This contains special words used in dBASE III PLUS with their definitions. A separate glossary for network systems can be found in *Networking with dBASE III PLUS*.

# Contents

## USING dBASE III PLUS


INTRODUCTION . . . . . I-1

Chapter 1: Overview . . . . . I-1
Chapter 2: Using Commands. . . . . I-1
Chapter 3: About Program Files . . . . . I-1
Chapter 4: dBASE III PLUS Configuration . . . . . I-1
Chapter 5: Commands. . . . . I-2
Chapter 6: Functions. . . . . I-2
Chapter 7: Error Messages . . . . . I-2
Appendices . . . . . I-2
Glossary . . . . . I-2
Index. . . . . I-3

CHAPTER 1
OVERVIEW. . . . . 1-1

dBASE III PLUS Technical Specifications . . . . . 1-1
Schematic of a Database Management System . . . . . 1-3
File Types . . . . . 1-4
    Binary (.bin) Files. . . . . 1-5
    Catalog (.cat) Files . . . . . 1-5
    Command and Procedure (.prg) Files . . . . . 1-5
    Database (.dbf) Files . . . . . 1-5
    Database File Backups (.bak) . . . . . 1-5
    Database Memo (.dbt) Files . . . . . 1-6
    Database Memo File Backups (.tbk) . . . . . 1-6
    Format (.fmt) Files . . . . . 1-6
    Index (.ndx) Files . . . . . 1-6
    Label (.lbl) Form . . . . . 1-7
    Memory (.mem) Files . . . . . 1-7
    Query (.qry) Files . . . . . 1-7
    Report Form (.frm) Files . . . . . 1-7
    Screen (.scr) Files . . . . . 1-7
    Text (.txt) Files . . . . . 1-7
    View (.vue) Files. . . . . 1-8
Database Structure . . . . . 1-8
    Character Fields . . . . . 1-9
    Date Fields . . . . . 1-9
    Numeric Fields. . . . . 1-9

Logical Fields . . . . . 1-9
Memo Fields . . . . . 1-9
Memory Variables (Memvars) . . . . . 1-10

**CHAPTER 2**
**USING COMMANDS . . . . . 2-1**

Entering Commands . . . . . 2-1
Re-entering Commands . . . . . 2-2
Structure of a dBASE Command (Syntax) . . . . . 2-3
Expressions . . . . . 2-4
Operators . . . . . 2-5
Precedence of Operators . . . . . 2-6
Basic Rules . . . . . 2-7
Using Memory Variables . . . . . 2-8
Memory Variable Data Types . . . . . 2-10
Using Functions . . . . . 2-11
Using Menus . . . . . 2-12
Menu Structure . . . . . 2-13
Navigating the Menus . . . . . 2-15
Selecting Menu Options . . . . . 2-16
Exiting Menus . . . . . 2-19
Using Screen Forms . . . . . 2-20
Using Commands for Your Database . . . . . 2-22
Creating a Database File . . . . . 2-22
Adding New Records . . . . . 2-23
APPEND, BROWSE, and INSERT . . . . . 2-23
APPEND BLANK and INSERT BLANK . . . . . 2-23
APPEND FROM . . . . . 2-23
Changing Records . . . . . 2-24
Changing Individual Records . . . . . 2-24
Changing a Batch of Records . . . . . 2-25
Deleting Records . . . . . 2-25
Organizing Database Files . . . . . 2-26
Indexing . . . . . 2-28
Sorting . . . . . 2-30
Searching for Data . . . . . 2-30

LOCATE and CONTINUE . . . . . 2-31
For Indexed Files: FIND and SEEK . . . . . 2-32
Using GOTO and SKIP . . . . . 2-32
Opening and Closing Files . . . . . 2-33
Number of Open Files . . . . . 2-33
Opening Files . . . . . 2-34
Closing Files . . . . . 2-35
Using Files Simultaneously . . . . . 2-36
Work Areas . . . . . 2-37
Aliases . . . . . 2-38
Linking Files with SET RELATION . . . . . 2-38
Storing a SET RELATION to a View (.vue) File . . . . . 2-41
Creating and Modifying a View File . . . . . 2-41
Displaying and Printing Data . . . . . 2-42
Reports and Labels . . . . . 2-42
CREATE/MODIFY REPORT . . . . . 2-42
CREATE/MODIFY LABEL . . . . . 2-43
Creating Screen Forms . . . . . 2-43
CREATE/MODIFY SCREEN . . . . . 2-44
The @ Command . . . . . 2-46
Opening Format Files . . . . . 2-46
Closing Format Files . . . . . 2-47
Printing Format Files . . . . . 2-47
Controlling Your Printer . . . . . 2-47
The Data Catalog . . . . . 2-49
Creating a Catalog . . . . . 2-51
Catalog Structure . . . . . 2-51
Opening a Catalog . . . . . 2-52
Adding Entries . . . . . 2-52
Deleting Entries . . . . . 2-53
Renaming Entries . . . . . 2-53
Closing Catalogs . . . . . 2-53
The Catalog Query (?) Clause . . . . . 2-54
Special Features of dBASE III PLUS . . . . . 2-55
Using Other Programs . . . . . 2-55
Merging Data from Two Files . . . . . 2-57
Calculating Totals . . . . . 2-58
Copying Database Files . . . . . 2-58

Copying Any Type of File (COPY FILE) . . . . . 2-59
Protecting Your Data . . . . . 2-59

**CHAPTER 3**
**ABOUT PROGRAM FILES . . . . . 3-1**

Why Do I Need Program Files? . . . . . 3-1
Programming Commands. . . . . 3-2
Structure of a Program File . . . . . 3-4
Memvars and Program Files. . . . . 3-6
Functions and Program Files . . . . . 3-6
Creating Program Files. . . . . 3-7
Executing Program Files . . . . . 3-8
Closing Program Files . . . . . 3-8
Debugging Program Files . . . . . 3-9
Some Advanced Features . . . . . 3-10
Procedure Files . . . . . 3-10
Other Features . . . . . 3-11

**CHAPTER 4**
**dBASE III PLUS CONFIGURATION . . . . . 4-1**

Config.sys . . . . . 4-1
Creating and Modifying Config.sys . . . . . 4-2
Config.db . . . . . 4-2
Configuration Commands. . . . . 4-4
SET Commands . . . . . 4-6
Function Keys . . . . . 4-7
Creating and Modifying Config.db . . . . . 4-7
256K Configuration . . . . . 4-10

**CHAPTER 5**
**COMMANDS . . . . . 5-1**

Symbols and Conventions . . . . . 5-1
Limitations . . . . . 5-2
Classes of Commands . . . . . 5-3
User Assistance. . . . . 5-3
Creation of Files . . . . . 5-4
Addition of Data . . . . . 5-4

Editing of Data . . . . . 5-5
Deletion of Data . . . . . 5-5
Data Display . . . . . 5-6
Positioning the Record Pointer . . . . . 5-6
Manipulating Database Files . . . . . 5-7
Manipulating Other Types of Files . . . . . 5-7
Using Memory Variables . . . . . 5-8
Programming . . . . . 5-9
Environmental Commands . . . . . 5-10
Parameter Controls . . . . . 5-10
External Program Interfacing . . . . . 5-13
Debugging Commands . . . . . 5-13
Event Processing . . . . . 5-14
?/?? . . . . . 5-15
@ . . . . . 5-16
@...TO . . . . . 5-23
ACCEPT . . . . . 5-25
APPEND . . . . . 5-26
APPEND FROM . . . . . 5-28
THE ASSISTANT . . . . . 5-32
AVERAGE . . . . . 5-39
BROWSE . . . . . 5-40
CALL . . . . . 5-43
CANCEL . . . . . 5-44
CHANGE . . . . . 5-45
CLEAR . . . . . 5-47
CLEAR ALL . . . . . 5-48
CLEAR FIELDS . . . . . 5-49
CLEAR GETS . . . . . 5-50
CLEAR MEMORY . . . . . 5-51
CLEAR TYPEAHEAD . . . . . 5-52
CLOSE . . . . . 5-53
CONTINUE . . . . . 5-54
COPY . . . . . 5-55
COPY FILE . . . . . 5-59
COPY STRUCTURE . . . . . 5-60
COPY STRUCTURE EXTENDED . . . . . 5-61
COUNT . . . . . 5-63

CREATE . . . . . 5-64
CREATE FROM . . . . . 5-65
CREATE/MODIFY LABEL . . . . . 5-66
CREATE/MODIFY QUERY . . . . . 5-70
CREATE/MODIFY REPORT . . . . . 5-73
CREATE/MODIFY SCREEN . . . . . 5-82
CREATE <newfile> /MODIFY STRUCTURE . . . . . 5-88
CREATE/MODIFY VIEW . . . . . 5-94
CREATE VIEW FROM ENVIRONMENT . . . . . 5-101
DELETE . . . . . 5-102
DIR . . . . . 5-103
DISPLAY . . . . . 5-105
DISPLAY HISTORY . . . . . 5-107
DISPLAY MEMORY . . . . . 5-108
DISPLAY STATUS . . . . . 5-109
DISPLAY STRUCTURE . . . . . 5-110
DO . . . . . 5-111
DO CASE . . . . . 5-113
DO WHILE . . . . . 5-114
EDIT . . . . . 5-118
EJECT . . . . . 5-120
ERASE . . . . . 5-121
EXPORT . . . . . 5-122
FIND . . . . . 5-124
GO/GOTO . . . . . 5-126
HELP . . . . . 5-127
IF . . . . . 5-128
IMPORT . . . . . 5-129
INDEX . . . . . 5-131
INPUT . . . . . 5-135
INSERT . . . . . 5-137
JOIN . . . . . 5-139
LABEL FORM . . . . . 5-143
LIST . . . . . 5-144
LIST HISTORY . . . . . 5-145
LIST MEMORY . . . . . 5-146
LIST STATUS . . . . . 5-147
LIST STRUCTURE . . . . . 5-148

LOAD . . . . . 5-149
LOCATE . . . . . 5-153
MODIFY COMMAND . . . . . 5-157
MODIFY LABEL . . . . . 5-158
MODIFY QUERY . . . . . 5-158
MODIFY REPORT . . . . . 5-158
MODIFY SCREEN . . . . . 5-158
MODIFY STRUCTURE . . . . . 5-158
MODIFY VIEW . . . . . 5-158
NOTE/*/&& . . . . . 5-159
ON . . . . . 5-160
PACK . . . . . 5-163
PARAMETERS . . . . . 5-164
PRIVATE . . . . . 5-165
PROCEDURE . . . . . 5-166
PUBLIC . . . . . 5-167
QUIT . . . . . 5-169
READ . . . . . 5-170
RECALL . . . . . 5-171
REINDEX . . . . . 5-172
RELEASE . . . . . 5-173
RENAME . . . . . 5-175
REPLACE . . . . . 5-176
REPORT . . . . . 5-178
RESTORE . . . . . 5-180
RESUME . . . . . 5-182
RETRY . . . . . 5-183
RETURN . . . . . 5-184
RUN/! <command> . . . . . 5-185
SAVE . . . . . 5-187
SEEK . . . . . 5-188
SELECT . . . . . 5-189
SET . . . . . 5-191
SET ALTERNATE . . . . . 5-193
SET BELL . . . . . 5-195
SET CARRY . . . . . 5-196
SET CATALOG . . . . . 5-197
SET CENTURY . . . . . 5-203

SET COLOR . . . . . 5-204
SET CONFIRM . . . . . 5-209
SET CONSOLE . . . . . 5-210
SET DATE. . . . . 5-211
SET DEBUG . . . . . 5-212
SET DECIMALS . . . . . 5-213
SET DEFAULT . . . . . 5-214
SET DELETED . . . . . 5-215
SET DELIMITERS . . . . . 5-216
SET DEVICE. . . . . 5-218
SET DOHISTORY . . . . . 5-219
SET ECHO . . . . . 5-220
SET ESCAPE. . . . . 5-221
SET EXACT . . . . . 5-222
SET FIELDS . . . . . 5-223
SET FILTER . . . . . 5-229
SET FIXED . . . . . 5-231
SET FORMAT . . . . . 5-232
SET FUNCTION . . . . . 5-234
SET HEADING . . . . . 5-235
SET HELP. . . . . 5-236
SET HISTORY . . . . . 5-237
SET INDEX . . . . . 5-239
SET INTENSITY . . . . . 5-241
SET MARGIN . . . . . 5-242
SET MEMOWIDTH . . . . . 5-243
SET MENU . . . . . 5-244
SET MESSAGE TO . . . . . 5-245
SET ODOMETER . . . . . 5-246
SET ORDER . . . . . 5-247
SET PATH. . . . . 5-249
SET PRINT . . . . . 5-251
SET PRINTER TO. . . . . 5-252
SET PROCEDURE. . . . . 5-253
SET RELATION . . . . . 5-255
SET SAFETY. . . . . 5-257
SET SCOREBOARD . . . . . 5-258
SET STATUS. . . . . 5-259

SET STEP . . . . . 5-260
SET TALK . . . . . 5-261
SET TITLE . . . . . 5-262
SET TYPEAHEAD TO . . . . . 5-263
SET UNIQUE . . . . . 5-264
SET VIEW TO . . . . . 5-266
SKIP . . . . . 5-268
SORT . . . . . 5-269
STORE . . . . . 5-271
SUM . . . . . 5-273
SUSPEND . . . . . 5-274
TEXT . . . . . 5-275
TOTAL . . . . . 5-276
TYPE . . . . . 5-278
UPDATE . . . . . 5-279
USE . . . . . 5-281
WAIT . . . . . 5-283
ZAP . . . . . 5-284

**CHAPTER 6**
**FUNCTIONS . . . . . 6-1**

Using Functions . . . . . 6-1
Input and Output Data Types . . . . . 6-2
Classification . . . . . 6-4
& . . . . . 6-7
ABS( ) . . . . . 6-10
ASC( ) . . . . . 6-11
AT( ) . . . . . 6-12
BOF( ) . . . . . 6-13
CDOW( ) . . . . . 6-14
CHR( ) . . . . . 6-15
CMONTH( ) . . . . . 6-17
COL( ) . . . . . 6-18
CTOD( ) . . . . . 6-19
DATE( ) . . . . . 6-20
DAY( ) . . . . . 6-21
DBF( ) . . . . . 6-22
DELETED( ) . . . . . 6-23

DISKSPACE() . . . . . 6-24
DOW() . . . . . 6-25
DTOC() . . . . . 6-26
EOF() . . . . . 6-27
ERROR() . . . . . 6-28
EXP() . . . . . 6-30
FIELD() . . . . . 6-31
FILE() . . . . . 6-33
FKLABEL() . . . . . 6-34
FKMAX() . . . . . 6-35
FOUND() . . . . . 6-36
GETENV() . . . . . 6-37
IIF() . . . . . 6-38
INKEY() . . . . . 6-40
INT() . . . . . 6-43
ISALPHA() . . . . . 6-44
ISCOLOR() . . . . . 6-45
ISLOWER() . . . . . 6-46
ISUPPER() . . . . . 6-47
LEFT() . . . . . 6-48
LEN() . . . . . 6-49
LOG() . . . . . 6-50
LOWER() . . . . . 6-51
LTRIM() . . . . . 6-52
LUPDATE() . . . . . 6-53
MAX() . . . . . 6-54
MESSAGE() . . . . . 6-55
MIN() . . . . . 6-56
MOD() . . . . . 6-57
MONTH() . . . . . 6-59
NDX() . . . . . 6-60
OS() . . . . . 6-61
PCOL() . . . . . 6-62
PROW() . . . . . 6-63
READKEY() . . . . . 6-64
RECCOUNT() . . . . . 6-67
RECNO() . . . . . 6-68
RECSIZE() . . . . . 6-69

REPLICATE() . . . . . 6-70
RIGHT() . . . . . 6-71
ROUND() . . . . . 6-72
ROW() . . . . . 6-73
RTRIM() . . . . . 6-74
SPACE() . . . . . 6-75
SQRT() . . . . . 6-76
STR() . . . . . 6-77
STUFF() . . . . . 6-79
SUBSTR() . . . . . 6-81
TIME() . . . . . 6-82
TRANSFORM() . . . . . 6-83
TRIM() . . . . . 6-84
TYPE() . . . . . 6-86
UPPER() . . . . . 6-88
VAL() . . . . . 6-89
VERSION() . . . . . 6-91
YEAR() . . . . . 6-92

**CHAPTER 7**
**ERROR MESSAGES AND THEIR EXPLANATIONS . . . . . 7-1**

**APPENDIX A**
**CURSOR CONTROL KEYS . . . . . A-1**

**APPENDIX B**
**BEGINNING/END OF FILE AND FILTER CONDITIONS . . . . . B-1**

Unindexed Files . . . . . B-1
Indexed Files . . . . . B-2
Filter Conditions . . . . . B-3

**APPENDIX C**
**STRUCTURE OF DATABASE (.DBF) AND MEMO (.DBT) FILES . . . . . C-1**

Structure of a Database File . . . . . C-1
The Header Structure . . . . . C-2
The Structure of Database (.dbf) File Records . . . . . C-2
The Structure of Database Memo (.dbt) Files . . . . . C-3

**APPENDIX D**
**ASCII CODE TABLES . . . . . D-1**

Printable Characters . . . . . D-1
Control Characters . . . . . D-3

**GLOSSARY . . . . . G-1**

# Chapter 1
## OVERVIEW

This chapter describes the technical specifications of dBASE III PLUS. As an overview, it also describes the different file types and field types, and discusses memory variables, which give you more flexibility in handling your data.

## dBASE III PLUS Technical Specifications

The following values may be limited by the amount of RAM in your computer.

**Database File**

| | |
|---|---|
| Number of records | — 1 billion maximum |
| Number of bytes | — 2 billion maximum |
| Record size | — 4,000 bytes in .dbf file<br>512 kilobytes in .dbt file |
| Fields | — 128 maximum |

**Field Sizes**

| | |
|---|---|
| Character fields | — 254 bytes maximum |
| Date fields | — 8 bytes |
| Logical fields | — 1 byte |
| Memo fields | — 5,000 bytes maximum or the capacity of the word processor used |
| Numeric fields | — 19 bytes maximum |

**File Operations**

15 open files of all types.

10 open database files. A database file counts as two files if memo fields are used.

Seven open index files per active database file.

One open format file per active database file.

## Numeric Accuracy

15.9 digits. Note that the decimal point does not count as a digit in determining accuracy. Accuracy is 13 digits when comparing numbers, except zero.

## Memory Variables

Number of active memory variables: 256 maximum.

Number of bytes available for memory variables: 6,000 default.

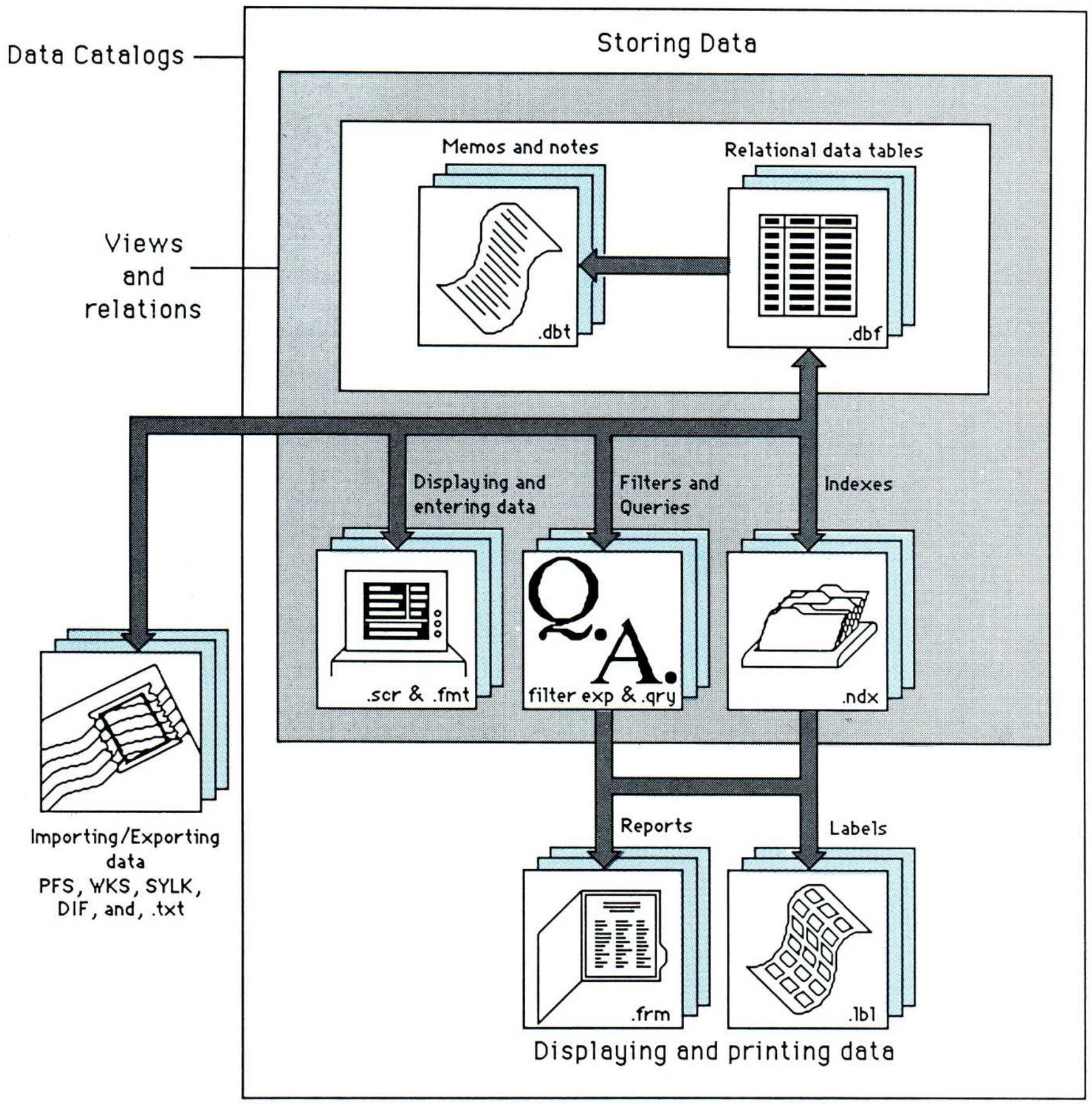
The size of a memory variable depends on its type:

| | |
|---|---|
| Character | — assigned string + 2 bytes |
| Numeric | — 9 bytes |
| Date | — 9 bytes |
| Logical | — 2 bytes |

## Command Line

Maximum length — 254 characters

# Schematic of a Database Management System



CHAPTER 1

## File Types

dBASE III PLUS saves information as disk files in several specialized formats. Each serves a specific dBASE III PLUS processing need.

Each disk file has a filename of not more than eight characters and a file extension consisting of a period and three characters maximum. A filename can contain letters, numbers, and the underscore (_) character, but not special characters or spaces. It cannot begin with a number.

You assign a filename at the time the file is created. The filename extension must be assigned at the same time. Normally, dBASE III PLUS assigns a default filename extension according to the type of file being created. However, you can override the dBASE III PLUS assignment by simply typing in an extension when you enter the filename. The default extensions, as well as the kinds of files, are defined in the following table:

| Kind of File | File Type |
|---|---|
| Binary | .bin |
| Catalog | .cat |
| Command and Procedure | .prg |
| Database | .dbf |
| Database Backup | .bak* |
| Database Memo | .dbt |
| Database Memo Backup | .tbk |
| Format | .fmt |
| Index | .ndx |
| Label | .lbl |
| Memory | .mem |
| Query | .qry |
| Report Form | .frm |
| Screen | .scr |
| Text Output | .txt |
| View | .vue |

* Command and procedure backup files also have a .bak extension.

Table 1-1 File types and extensions

If you do not use the default extension, you must include the extension whenever the file is selected for use.

## Binary (.bin) Files

Binary files are assembly language programs that are assembled, linked, and converted to a binary form so they can be LOADed directly into memory and executed with the CALL command. Binary files are created when an assembled and linked program is converted from the .EXE format to the .COM format with the DOS EXE2BIN command.

## Catalog (.cat) Files

A catalog (.cat) file (or catalog, for short) is like a register to help you keep track of all your files. For example, if you have a database file that you use regularly with an index file, a custom screen format, a report form, a view file, and a query file, you can store the names of all these files in a catalog. When you activate the catalog, only these files can be opened. A catalog file is created and activated with the SET CATALOG TO command.

## Command and Procedure (.prg) Files

Command files contain sets of dBASE III PLUS commands that have been stored as programs. They are ASCII files and may be created with either MODIFY COMMAND or any word processing program that creates ASCII text files.

Procedure files also contain dBASE III PLUS instructions, but they are stored in memory. A procedure file is like a programming tool kit. It contains utility programs (called procedures) which complete simple tasks that you do often in dBASE III PLUS. Having a procedure file in memory reduces disk access time and enables an applications program to run faster. You can only have one procedure file in memory at a time.

## Database (.dbf) Files

Database (.dbf) files store data in records and fields. Viewing a database file as a table, each record represents a row in the table and each field a column. Each record is intended to contain a set of unique information. The structure of records in a database file is created with CREATE and changed with MODIFY STRUCTURE.

## Database File Backups (.bak)

Each time you modify the structure of a database file, dBASE III PLUS creates a backup of the file, and a new file containing the new structure. The data in the backup file is added to the new file in fields with the same name.

Back-up files with a .bak extension are also created when you alter a file that was created with MODIFY COMMAND.

## Database Memo (.dbt) Files

Database memo (.dbt) files are text files which contain the contents of memo fields. All memo fields in a database (.dbf) file are stored in the same database memo file. The .dbt file is created at the same time as the .dbf file.

Because the contents of memo fields are stored in a separate file, a memo field only occupies ten spaces per record in the database file.

## Database Memo File Backups (.tbk)

If a database file has memo fields, they are stored in a memo (.dbt) file. When you modify the structure of the database file, a new database file with the new structure as well as a backup of the memo file is created and saved. The data in the back-up file is added to the new file in memo fields with the same name.

## Format (.fmt) Files

You use format files to create custom screen forms for use in data entry and printed output. They are generated from screen (.scr) files set up by the CREATE SCREEN command. You can also write format files directly using MODIFY COMMAND or any word processor that creates ASCII text files.

## Index (.ndx) Files

Index files enable you to use a database file in a logical order rather than the physical order. The physical order is the order in which you enter records in a database file. A logical order is an alphabetical, chronological, or numerical order based on the contents of one or more selected fields. The selected field is called a key field. Index files are created with the INDEX command.

Index files store the database record numbers according to the selected field(s). When using a database with an index file, the database appears to be in the order of the key item. For example, if the key field is a name field, the records appear in alphabetical order according to the names.

The key, which can be one or more fields, may also be used to provide direct access to a particular record.

**Label (.lbl) Form**

Label files contain the information needed by the LABEL command to print labels. They are created and altered by the CREATE/MODIFY LABEL command.

**Memory (.mem) Files**

Memory files are used to save the contents of memory variables for later use. Memory files may contain up to 256 memory variables or 6,000 bytes (default value). These files are created by the command SAVE and are read back into memory by the command RESTORE.

**Query (.qry) Files**

Query files contain a filter condition that restricts the records that are displayed with commands that require a database (.dbf) file to be in USE. Query files are created with the CREATE/MODIFY QUERY command.

**Report Form (.frm) Files**

Report form files contain the information needed by the REPORT command to prepare reports. They can be created and altered by the CREATE/MODIFY REPORT command.

**Screen (.scr) Files**

Screen files contain information that is used by the CREATE/MODIFY SCREEN command to generate and modify screen files. A companion format (.fmt) file is automatically created when you use these commands.

**Text (.txt) Files**

Text files are primarily used to interface dBASE III PLUS with software that reads ASCII files. These files should contain only printable ASCII characters.

You read text files from other software into dBASE III PLUS with a special form of the APPEND FROM command. You create text files from dBASE III PLUS files for use with other software via a special form of the COPY command. A text file is also created when you use the SET ALTERNATE command to store screen output such as the commands you type at the dot prompt and their results or the output from a program file.

## View (.vue) Files

A view file contains a list of selected field names from one or more specified database (.dbf) files. You set up a view file so only the selected fields are seen when using the database files. View files can also store associated index files and relationships between the database files. You can create a format file and filter conditions which reference the selected field names. You use CREATE/MODIFY VIEW to set up the view file. A view file can also be stored in a catalog.

## Database Structure

The structure of a database file is established by defining fields to store data. You use the CREATE and MODIFY STRUCTURE commands to define a field which consists of the name, type, and width.

Field names may be up to ten characters long. They must begin with a letter and cannot contain embedded blank spaces. Letters, numbers, and underscores are permitted.

There are five field types in dBASE III PLUS. They are Character (C), Date (D), Logical (L), Memo (M), and Numeric (N). Each one is discussed below.

The field width is the maximum number of characters or digits that are to be contained in the field. You define the field width for character and numeric fields. Date, logical, and memo fields have fixed lengths. In the case of numeric fields, you need to include the sign (if any), the decimal point, and the number of decimal places (digits to the right of the decimal point) in the total number of characters. See Appendix C for further information.

## Character Fields

Character fields may be used to store any printable ASCII character (letters, numbers, special symbols, and blank spaces). The maximum field width of a character field is 254.

## Date Fields

Date fields are used to store dates. Dates may be entered and displayed in different ways by using the SET DATE command or the PICTURE option of the @ command. The default date entry and display format is mm/dd/yy. The field width of a date field is always eight.

Dates may be used in calculations with other dates and numbers. For example, a number may be added to a date to calculate a date in the future; a date may be subtracted from a date; and a number may be subtracted from a date.

## Numeric Fields

Numeric fields are of two types: integer and decimal. An integer is a number that does not have decimal places (for example, the number of students in a class). The width of a numeric field is the number of digits the field can hold. The decimal point and negative sign each count as one digit.

Accuracy of numeric fields extends to 15.9 digits, not including the decimal point itself. This means that the most significant 15 digits of any number will be reliable.

## Logical Fields

Logical fields accept only single characters representing True/ False values. Logical True (.T.) is entered as T, t, Y, or y. Logical False (.F.) is entered as F, f, N, or n.

## Memo Fields

Memo fields are designed to accommodate large blocks of textual information. All memo fields in a database file are stored in a database memo (.dbt) file which is created when you set up the database file. Memo fields are indicated in the database (.dbf) file by the word *memo*.

Memo fields may contain the same kind of information as character fields, but can have 5,000 characters or more, depending on the word processor you use. Each memo field requires a minimum of 512 bytes per record if any data is specified. Each memo field occupies only ten spaces per record in the database file, because the text is stored in a separate file on your disk.

## Memory Variables (Memvars)

Memory variables enable you to store data outside of the database file structure. For example, the contents of a field in several records can all be summed, and the result stored as a memory variable. The content of the memory variable could then be directly used in further calculations. Memory variables are often used in programming applications where they provide a means for controlling the program. Memory variables ordinarily provide temporary storage, and their values are kept only during a current session in dBASE III PLUS. However, you can SAVE memory variables to a memory (.mem) file, and RESTORE them to reuse their values.

Here are the specifications for memory variables:

### Memvar Names

Memory variable names can be up to ten characters long. They can contain letters, numbers, and underscores. They must begin with a letter and cannot contain embedded blank spaces.

The only restricted names for memory variables are those created with the syntax < memory variable > = < expression >. In this case, the memory variable name cannot be the same as a dBASE III PLUS command.

### Memvar Types

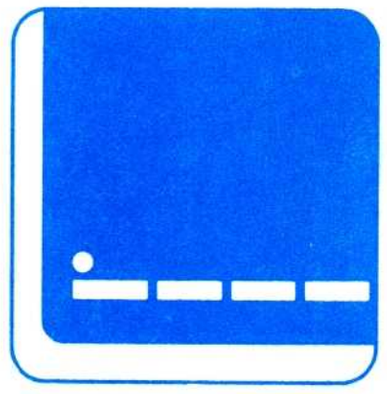
There are four types of memory variables: character, date, numeric, and logical.

### Memvar Sizes

In memory, the size of a memory variable depends on its type. Character type variables are the length of the assigned string plus two bytes. Date variables are nine bytes. Numeric variables are also nine bytes. Logical variables take up two bytes.

There can be as many as 256 active memory variables with a combined size of up to 6,000 bytes. This 6,000-byte capacity can be increased by using the MVARSIZ option in Config.db, provided that you have more memory than the recommended 384K. (See Chapter 4 in this manual for more information on Config.db.)

# Chapter 2
## USING COMMANDS



This chapter tells you how to enter commands, describes the structure of a dBASE III PLUS command, and demonstrates how to manipulate your files using the different commands.

You'll find out which commands to use to accomplish several data handling tasks. There are commands to create, change, edit, organize, search, and print your database files. There are commands to design custom screen forms and print them. There are also commands which enable you to use several database files at one time. Additional commands let you use your files from other programs with dBASE III PLUS.

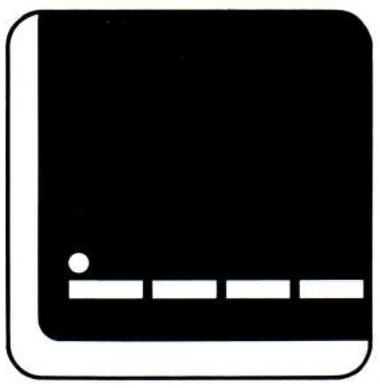
You may read this chapter from beginning to end to get an overview of the dBASE III PLUS commands. Or, if you are looking for specific information on how to complete a task, browse through the table of contents or through the chapter. The sections are task-oriented, designed to help you find the commands you need.

> **TIP**
> This chapter provides working examples of dBASE III PLUS commands using the Sample Programs and Utilities Disk. You should work with an unused copy of the original disk to ensure that the results you obtain when running these examples match the results shown.

## Entering Commands

This section tells you how to enter commands at the dot prompt and describes the structure of a dBASE command, which you can tailor to meet a particular need.

You issue a command in dBASE III PLUS by typing the command at the dot prompt and pressing ↵. While you're typing the command, you can use the navigation and editing keys listed in Table 2-1 to make corrections and changes:





| Key | Function |
|---|---|
| → | Move cursor one character to right. |
| ← | Move cursor one character to left. |
| ↑ | Move cursor to beginning of previous command (see next section). |
| ↓ | Move cursor to beginning of next command. |
| **Home** | Move cursor to previous word in command. |
| **End** | Move cursor to next word in command. |
| **Backspace** | Delete one character to left of cursor. |
| **Del** | Delete one character at cursor position. |
| **Ins** | Switch modes between inserting and over-writing characters. |
| **Num Lock** | Turn Numeric mode on and off. Must be off for function keys on numeric keypad to operate. |
| **Caps Lock** | Switch between upper case and lower case modes. |

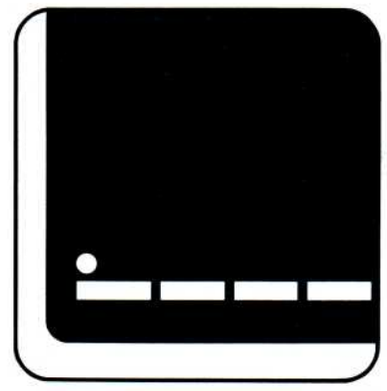
Table 2-1 dBASE navigation and editing keys

## Re-entering Commands

The dBASE III PLUS dot prompt has a memory buffer called *history* that automatically stores the last 20 commands you enter. This allows you to go back and edit or run a previously entered command. You can use the SET HISTORY TO command to change the default number of stored commands from 20 to any number you prefer.

Press ↑ at the dot prompt to display commands previously stored in the history buffer. The commands appear one at a time in reverse order; that is, the last command you entered appears first, the next-to-last command appears second, and so on. ↓ moves the cursor back down the list of commands. Use DISPLAY HISTORY and LIST HISTORY to view more than one command at a time.

To run a command stored in the history buffer, display the command at the dot prompt as just described and then press ↵. To edit a command in the buffer, display the command at the dot prompt, then use the editing keys listed in the table above to make any necessary changes. Press ↵ to run the new command.



## Structure of a dBASE Command (Syntax)

The structure of a command is called its syntax. Each command line begins with a verb, which is the basic command. Many command lines also have one or more clauses that tailor the command to meet a particular need. The general syntax of a command is illustrated by:

**VERB [ < scope > ] [ < expression list > ] [FOR < condition > ] [WHILE < condition > ]**

The verb is the name of the dBASE command.

The [ ] (square brackets) indicate that the item is optional.

The < > means that you enter the enclosed item.

< scope > defines how many records the command applies to. The keywords for scope are: RECORD < n > to specify a single record by its number; NEXT < n > means n records beginning with the current record; ALL for all the records in the database; and REST for records from the current record to the end of the file.

< expression list > is one or more expressions, separated by a comma. They do not have to be the same data type (see Expressions below).

FOR < condition > is known as a clause or condition statement. It tells dBASE III PLUS that the command applies only to records which meet the condition. WHILE < condition > tells dBASE III PLUS to repeat the command as long as long as the condition is true.

< condition > is a comparison between two items like Name = 'Smith' or a logical statement like .NOT. EOF().

Do not type the square brackets ([ ]) or angle brackets ( < > ) when entering a command.

To illustrate the syntax of a typical command:

```
DISPLAY NEXT 25 Name,Address FOR Name='Jones'
```

verb — scope — expressions list — condition

The specific syntax of each command is shown in Chapter 5.

## Expressions

An expression is formed with any combination of:

Fields
Memory Variables
Constants
Functions
Operators

For example, the Tours database contains the numeric fields Unitcost, Percent, and Seats_open. To find the cash value of the commissions available on unsold seats for a particular tour, use the following command:

```
. ? (Unitcost * Seats_open) * Percent/100
```

(Unitcost * Seats_open) is a numeric expression. dBASE III PLUS accepts several types of expressions, all of which are outlined below:

| Expression Type | Abbreviation |
|---|---|
| Character | expC |
| Date | expD |
| Logical | condition |
| Numeric | expN |

When you combine fields or memory variables or both in one expression, they must all be the same data type. If necessary, use functions to convert them. Note, however, that an expression list does not have to be one data type.

When you use functions to convert data types of fields and memory variables, you do not modify the fields in the database file, nor do you modify the structure of the database file. The original data type of a memory variable also remains unchanged.

For more information on the functions which convert data types, see the classification in Chapter 6, "Functions," and Chapters 5 and 7 in *Programming with dBASE III PLUS*.

## Operators

dBASE III PLUS provides four types of operators: mathematical, relational, logical, and string.

### Mathematical Operators

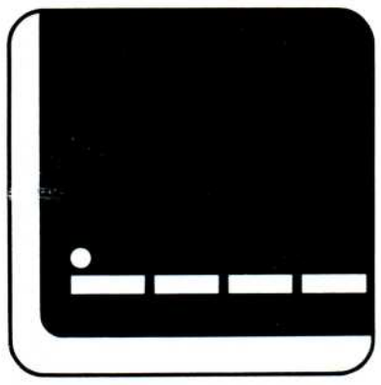
Mathematical operators generate numeric results.

| | |
|---|---|
| + | Addition |
| − | Subtraction |
| * | Multiplication |
| / | Division |
| ** or ^ | Exponentiation |
| ( ) | Grouping |

### Relational Operators

Relational operators generate logical results (for example, True or False). Relational operators can be used with either character, numeric, or date expressions. Both expressions must be of the same type.

| | |
|---|---|
| < | Less than |
| > | Greater than |
| = | Equal |
| < > or # | Not equal |
| < = | Less than or equal |
| > = | Greater than or equal |
| $ | Substring comparison (For example, if A and B are character strings, A$B returns a logical True if A is either identical to B or contained within B) |





### Logical Operators

Logical operators obtain a logical result from comparing two logical expressions.

| | |
|---|---|
| .AND. | Logical and |
| .OR. | Logical or |
| .NOT. | Logical not (works with a single expression) |
| ( ) | Parentheses for grouping |

### String Operators

| | |
|---|---|
| + | Concatenation operator. It is used to join two or more character strings into a single character string. |
| − | Concatenation operator. It is used to join two or more character strings into a single character string. Trailing spaces of the string preceding the operator are moved to the end of the string following the operator. |

## Precedence of Operators

Each type of operator has a set of rules that governs the order in which operations are performed. These are referred to as the *order of precedence* for the operator.

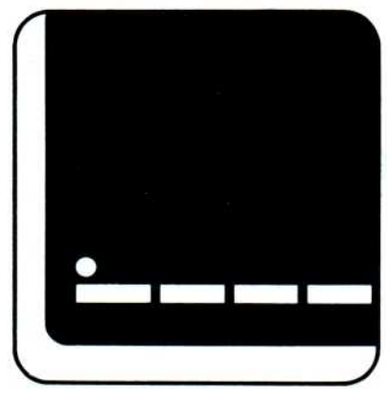
The relational and string operators have only one level of precedence and are performed in order from left to right.

The precedence levels for mathematical operators are:

1. Unary + and − signs
2. Exponentiation
3. Multiplication and division
4. Addition and subtraction

The precedence levels for the logical operators are:

1. .NOT.
2. .AND.
3. .OR.



When several of the four types of operators are used in the same expression, the precedence levels for the types are:

1. Mathematical and string
2. Relational
3. Logical

All operations at the same precedence level are performed in order from left to right. Parentheses are used to override the order in which operations are performed. Operations within the innermost nested parentheses are performed first.

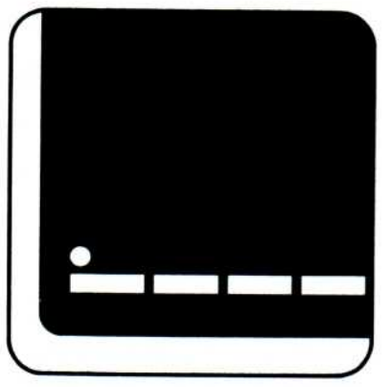
## Basic Rules

When using the dBASE III PLUS command language, there are certain rules that must be followed to ensure that commands are properly written:

1. Each command must begin with a command verb taken from Chapter 5 in this manual.
2. Each command must conform to the syntax for that command as described in Chapter 5.
3. A command consists of a verb and, optionally, one or more clauses and keywords that qualify the verb. Optional clauses (beginning with WHILE or FOR) and keywords such as NEXT may occur in any order. For example, the following are equivalent:

```
DISPLAY NEXT 25 FOR Name = 'Johnson'
DISPLAY FOR Name = 'Johnson' NEXT 25
```

4. The maximum command line length is 254 characters.
5. Words within a command may be separated by any number of blank spaces. Blank spaces, however, count in the 254-character limit.
6. Commands and certain keywords may be abbreviated to the first four characters, for example, DISPLAY MEMORY can be abbreviated as DISP MEMO.





7. Command verbs, keywords, field names, memory variable names, and filenames may be in upper case, lower case, or any mixture.

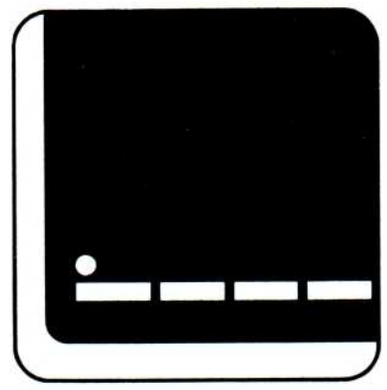
8. Do not use the letters A through J and M by themselves as database filenames because they are reserved for ALIAS names. For example, "AA" is a valid database filename, whereas "A" is not. Also, do not use DOS device names as filenames because these are reserved by the operating system. See your DOS manual for details.

9. Although dBASE III PLUS has no prohibited words, try to avoid using keywords and commands for filenames, field names, or memory variable names, as this may cause some confusion. For example, a field named Status could not be displayed by the command DISPLAY STATUS because the command would display the current processing status rather than the content of the field Status.

## Using Memory Variables

Memory variables are data values stored temporarily in memory. You use these values to perform calculations, comparisons, and other operations on your database files. Memory variables can be created by any of the following commands: ACCEPT, COUNT, PARAMETERS, PUBLIC, SUM, AVERAGE, INPUT, STORE, and WAIT.

You assign each memory variable a name of up to ten characters long, including letters, numbers, and underscores. The name must begin with a letter and cannot contain embedded blank spaces. The only restricted names for variables are those created with the syntax < memvar name > = < expression >. In this case, the memory variable name cannot be the same as a dBASE III PLUS command.

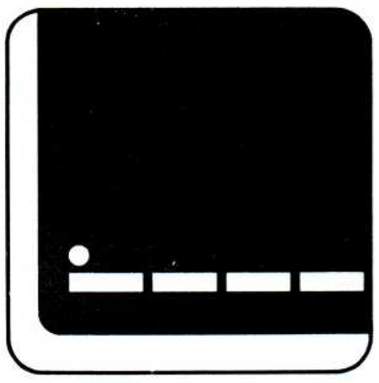


**NOTE**
When fields and memory variables have the same name, fields take precedence over memory variables. You can change this by preceding the memory variable name with m->, as in *m->memvar*. Another way is to use the & function, which can only be used with a memory variable. To use the & function, you add the & symbol to the memory variable name, for example, *&memvar*.

Variables are normally changed by overwriting existing variables with the same name. The only command that can edit the content of an existing variable is @...GET.

You can also combine memory variables with fields or other memory variables to form expressions. Each item in the expression must be the same data type. If necessary, use the appropriate functions to convert them. (See Chapter 6 in this manual.) Remember that converting the data type does not change field types in the database, nor does it modify the structure of a database. The original data type of the memory variable also remains unchanged. The conversion is only used in the operation being performed.

Memory variables provide temporary storage of data; their contents are only available during the current dBASE III PLUS session. However, they can be saved in a memory (.mem) file by using the SAVE command. Variables can be reused from a memory file by means of the RESTORE command.



CHAPTER 2

# Memory Variable Data Types

There are four data types for memory variables: character, numeric, date, and logical. These are discussed below.

## Character

Character type memory variables are normally used to store character strings. However, a binary sequence may be stored to a character string by means of the CHR( ) function. Character variables may contain as many as 254 characters. The total memory space used by a character variable is its length in bytes plus two bytes.

## Date

Date type memory variables are used to store dates. The size of a date variable is always eight bytes. The total memory requirement is nine bytes. Date variables are validated whenever they are entered or changed. They are normally presented in the mm/dd/yy (American) format. They may be presented in other formats by use of the SET DATE and SET CENTURY commands.

Date variables may be created from other date variables or fields. The CTOD( ) function must be used when you enter a date directly from the keyboard.

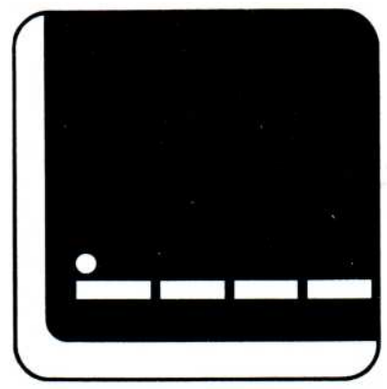
A date may be subtracted from another date. The result is a number (the number of days between the dates). A number (representing a number of days) may be added to or subtracted from a date. The result will also be a date.

## Numeric

Numeric type memory variables are used to store numbers that may be used in calculations. Numeric variables use up to nine bytes in memory. The accuracy of numeric variables extends to 15.9 digits. This means that the most significant 15 digits are reliable (not counting the decimal point). This is reduced to 13 digits for comparisons of non-zero numbers.

## Logical

Logical variables are stored as True (.T.) or False (.F.). The size of a logical variable is always one byte. It requires a total of two bytes of memory. Logical variables will accept T, t, Y, or y for True and F, f, N, or n for False. When created from the keyboard, a logical value must be delimited by periods, for example, STORE .T. TO Mlogic.



## Using Functions

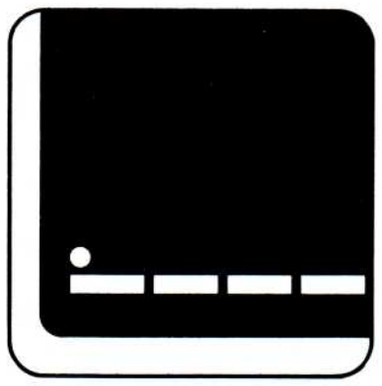
Functions are used to evaluate data and *return a result*. They enhance your use of the dBASE III PLUS commands; in fact, functions are always used with a command, and never alone.

dBASE III PLUS functions all have () -- opening and closing parentheses -- after the function name. The () may or may not contain data to be evaluated.

> **NOTE**
> The & function (also known as the macro function) is the only function which does not use parentheses. Instead, they are replaced with a memory variable name, for example, *&mname*. See Chapter 6 for more information on the & function.

The following examples shows several functions with the **?** (query) command.

```
. ? RECNO()
3
. ? RECNO()+3
6
. ? EOF()
.F.
. ? CHR(54)
6
. ? ABS(-100)
100
. ? SORT(144)
12.00
. STORE 'rick' TO name
rick
. ? LEN(name)
4
. ? UPPER(name)
RICK
```





In these examples, RECNO() checks the location of the record pointer and displays the record number; EOF() checks if the record pointer is at the end of a file; CHR() displays the ASCII value of a character; ABS() displays the absolute value of a number; SQRT() displays the square root of a number; LEN() displays the number of character in a character expression; UPPER() converts lower-case letters to upper-case letters

In expressions, functions are very useful for converting data types of fields and memory variables. Remember that items in an expression must be one data type (see Expressions section above). When you use functions to convert data types of fields and memory variables, you do not modify the fields in the database file, nor do you modify the structure of the database file. The original data type of a memory variable also remains unchanged.

## Using Menus

There are several commands in dBASE III PLUS which are menu-driven; that is, you enter the command at the dot prompt and a menu appears with several options to choose from. Each option presents a task you can do with that menu.

All dBASE III PLUS menus use a standard interface. This section describes the underlying structure of menus and navigation keys, and how to work your way through any command that uses the menu system. These are:

- ASSIST
- BROWSE
- CREATE
- CREATE/MODIFY LABEL
- CREATE/MODIFY QUERY
- CREATE/MODIFY REPORT
- CREATE/MODIFY SCREEN
- CREATE/MODIFY VIEW
- MODIFY STRUCTURE
- SET

## Menu Structure

Study the following screen. Each part of the menu is explained below.

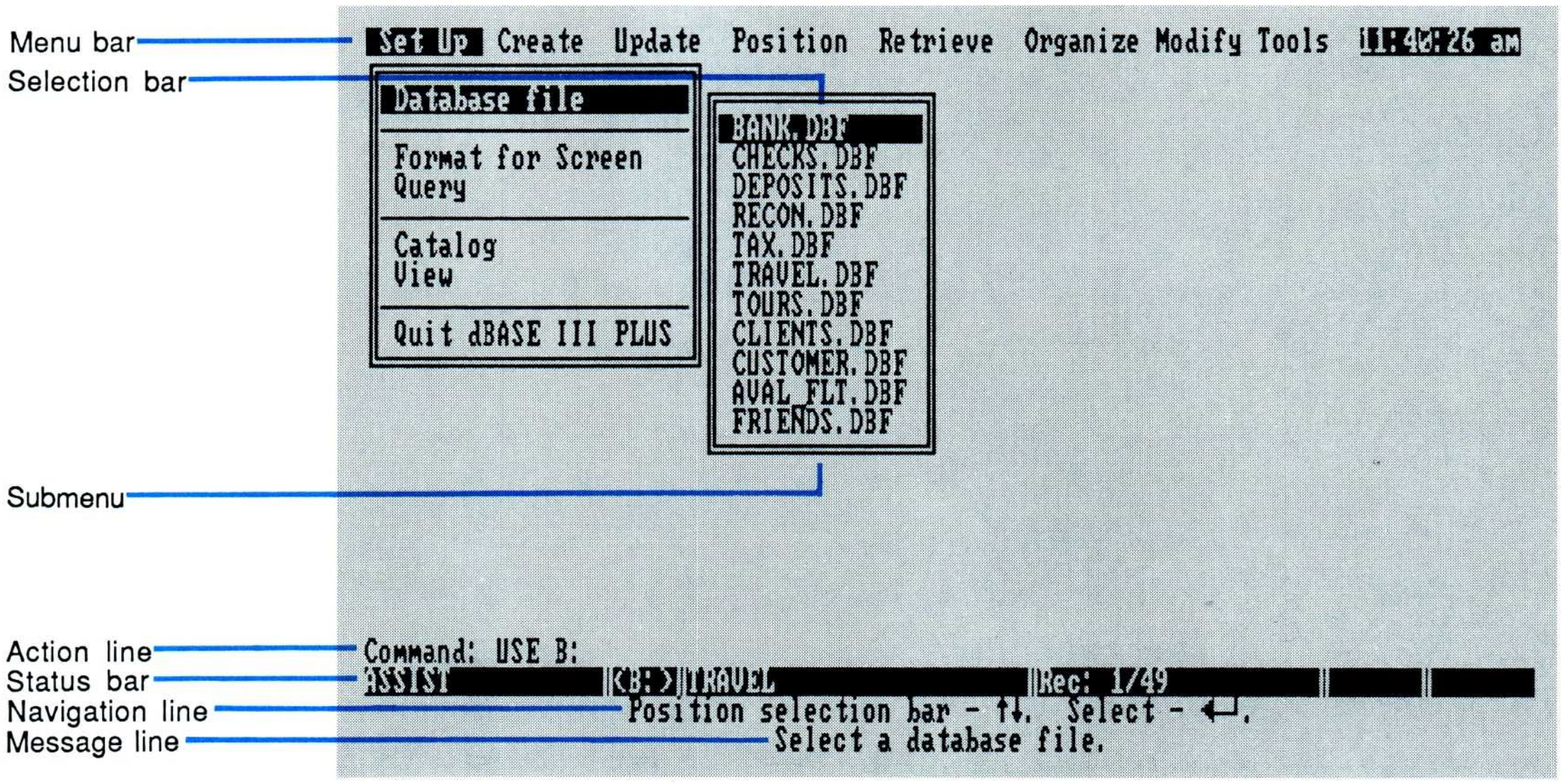



Figure 2-1 The dBASE III PLUS menu screen

### Menu Bar

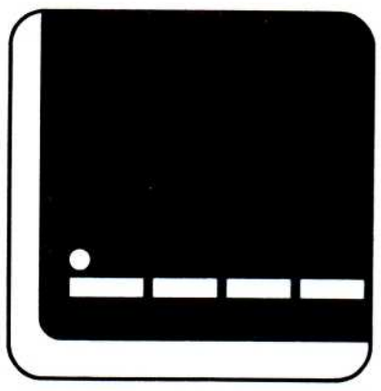
The menu bar is located at the top of the screen and contains several menus and a clock. Some commands are menu-driven, meaning that the command is operated using the menu bar. Other commands are menu-assisted, meaning that the menu bar is optional. In these commands, you will not see a menu bar unless you ask for it by pressing **Ctrl-Home** or **F10**.

### Main Menu

Each menu on the menu bar has a main pull-down menu associated with it.

### Selection Bar (Highlight)

The selection bar or highlight is a reverse video bar that indicates the current menu option. Several selection bars may be on the screen at one time. For example, when using The Assistant, selecting the Database file option under Set Up presents a submenu of drive letters. You can see that **Database file** is still in reverse video. Using the ↑ and ↓ keys moves the selection bar in the drive letter submenu until you make a selection by pressing ↵.





**Submenu** Some main menu options activate a submenu of further options from which you make selections. A submenu is similar to a main menu in that it offers a set of options. An option can, in turn, call another submenu. A submenu can be a list of things such as fields, files, or drive letters, from which one, or sometimes more than one, option can be selected.

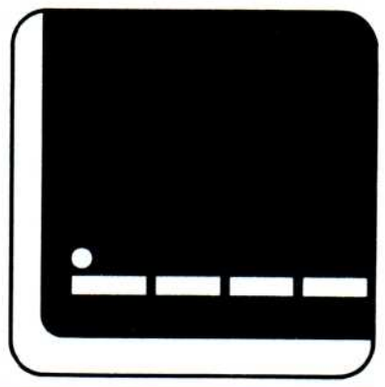
**Dialogue Box** This box displays long prompts and allows you to enter multiple parameters to perform an operation. For example, when creating an index, you are prompted to enter the key expression in the dialogue box.

**Action Line** As you build a command in The Assistant, the syntax is written on the action line. This helps you learn the dBASE III PLUS syntax of individual commands so you can build them from the dot prompt or in a program. Other menu-driven commands, such as CREATE/MODIFY VIEW, use this line to provide useful information.

**Status Bar** The status bar provides pertinent information about the current operation: the name of the operation, the active filename, the current drive, the number of options in a menu and which option is highlighted, the number of records in a file, and the record at which you're positioned. The status bar also indicates whether you have pressed the **Cap Lock**, **Ins**, or **NumLock** keys.

**Navigation Line** The navigation line provides instructions for moving around the screen and selecting an option, as well as displaying error messages in case of a mistake.

**Message Line** The message line explains the active menu option.



## Navigating the Menus

The selection bar moves horizontally along the menu bar. The navigation keys are:

| Key | Movement |
|---|---|
| → | Right one menu |
| **End** | Last menu on right |
| ← | Left one menu |
| **Home** | First menu on left |
| <a – z> | Next menu to right with matching initial letter |

Table 2-2 Navigating along the menu bar

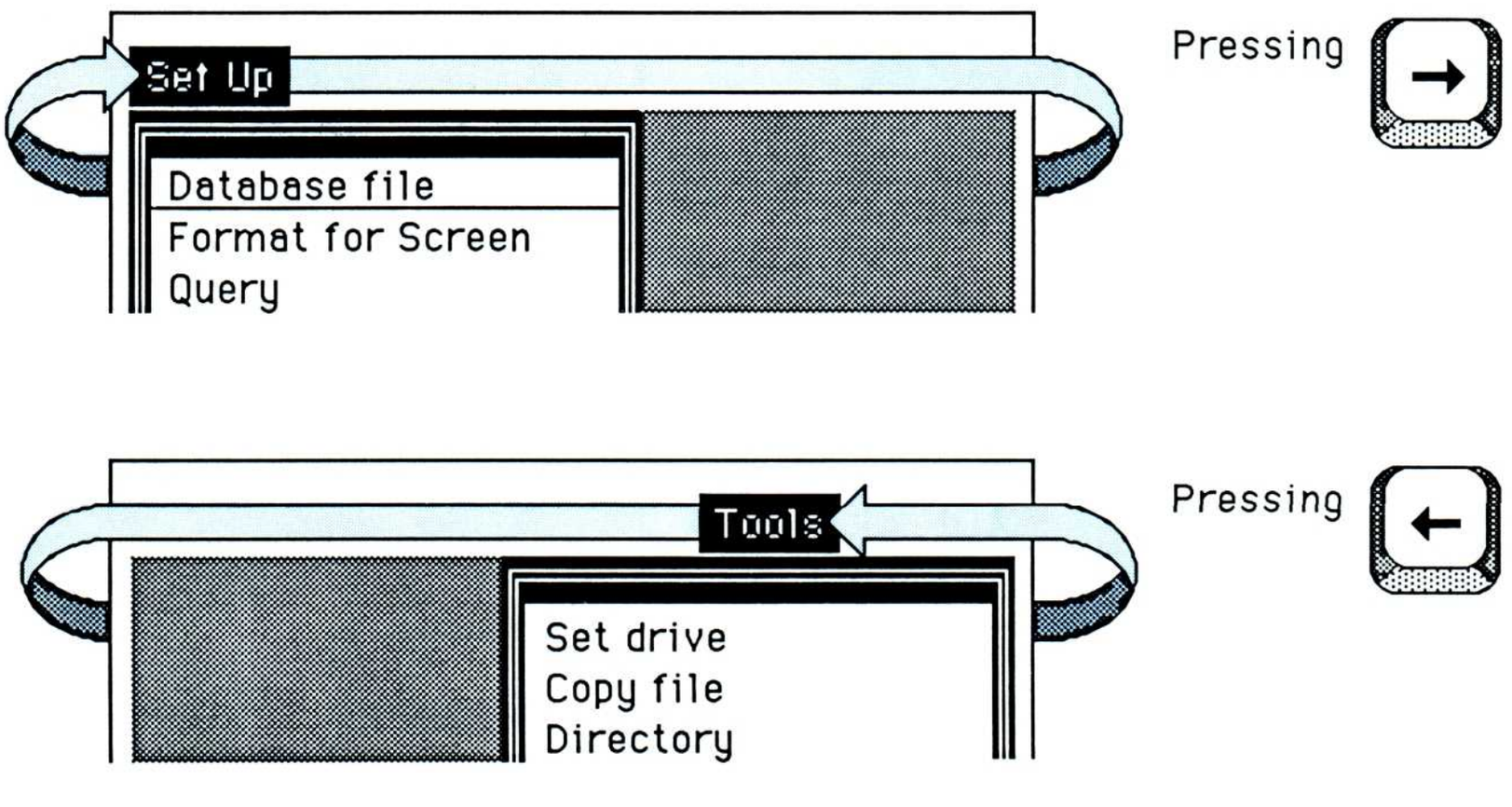



Figure 2-2 Moving horizontally

The selection bar moves vertically around the options on the pull-down menus (main menus and submenus alike). The navigation keys are:

| Key | Movement |
|---|---|
| ↓<br>**PgDn** | Down one option; at bottom option moves to top<br>Bottom option |
| ↑<br>**PgUp** | Up one option; at top option moves to bottom<br>Top option |

Table 2-3 Navigating on the pull-down menus

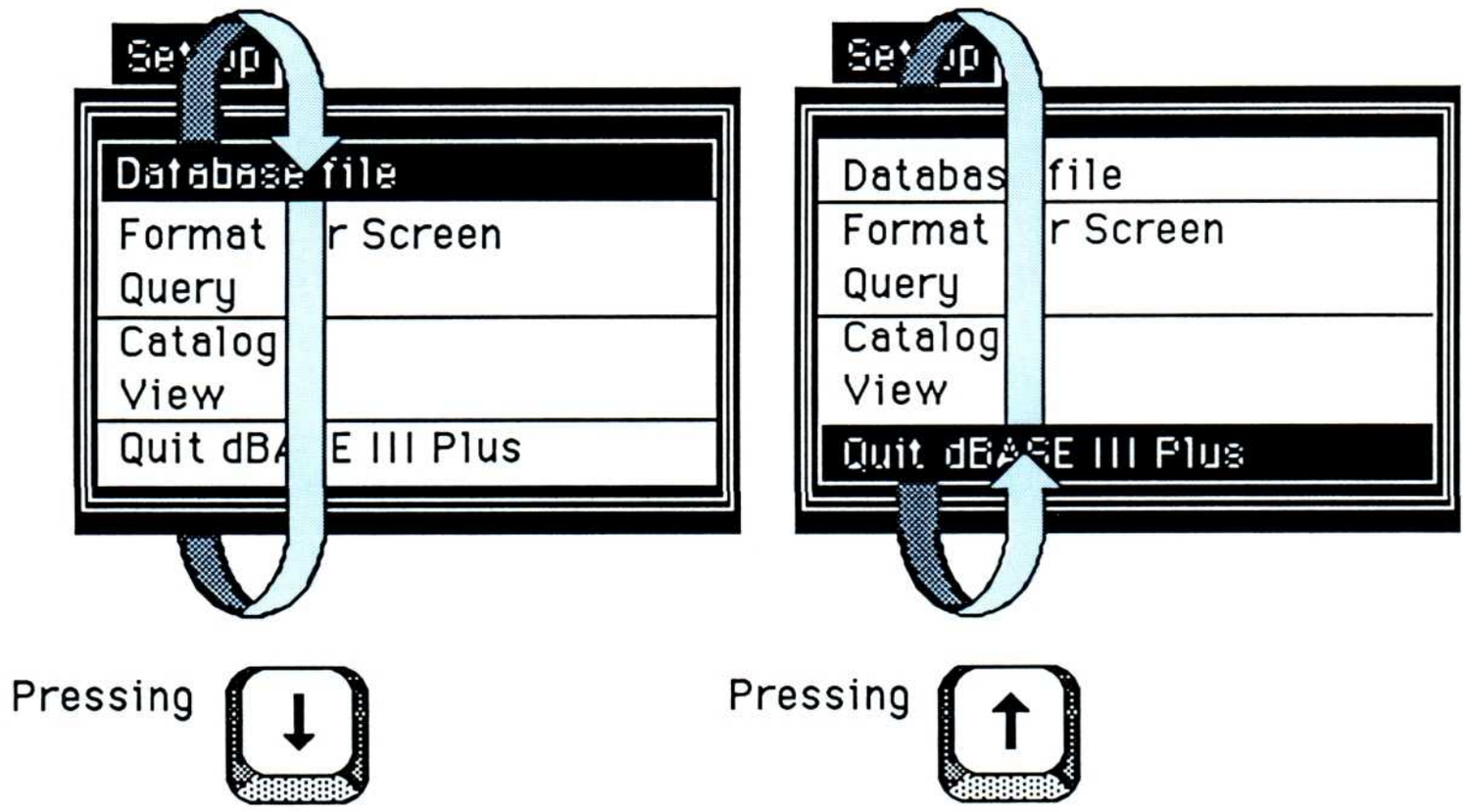
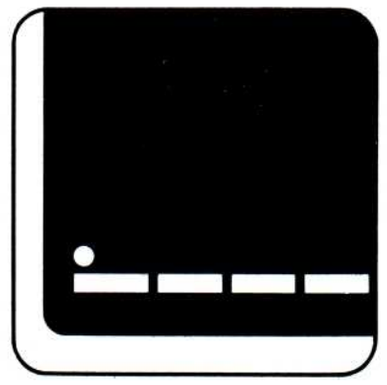



Figure 2-3 Moving vertically

## Selecting Menu Options

There are three types of menu options, explained below. When the selection bar is on the option you want, press ↵ to select that option. Then, depending on the option you've selected, you can make additional selections, answer Yes or No, or type an entry.



**Enumerated Values**

Some menu selections have only a small number of possible values, referred to as enumerated values. You scroll through the set of possible values by pressing ←┘. When the value in the menu matches what you want, use the arrow keys to go on to another option. An example of this type of a menu option can be found in the SET menu displayed in Figure 2-4. In its Options submenu, you press ←┘ to toggle an option ON or OFF. In the Screen submenu, you press ←┘ to scroll through different screen colors until the color you want appears.

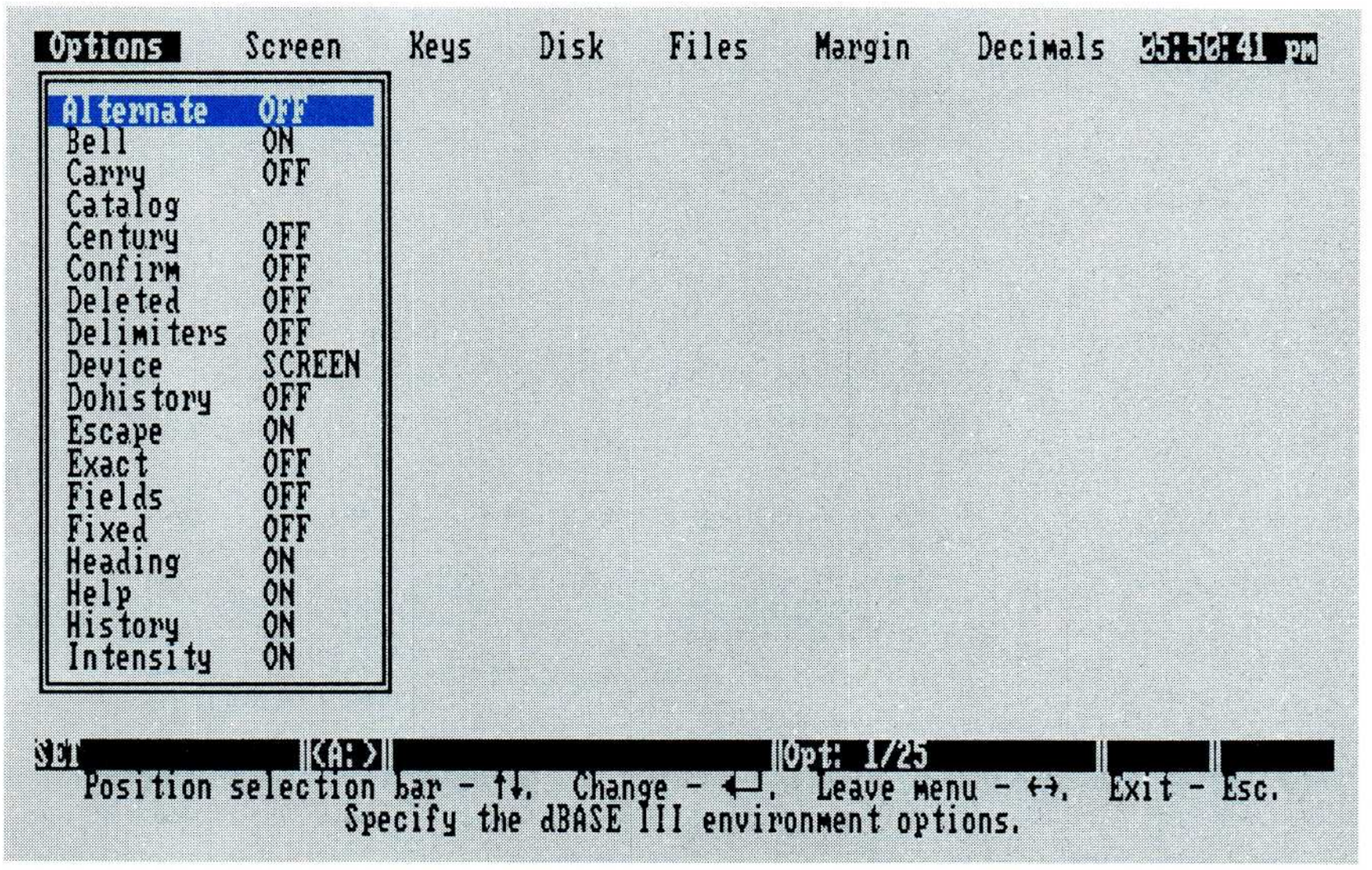



Figure 2-4 Enumerated value selections

**Submenu Lists**

Some menu selections activate a submenu as in Figure 2-5. Submenus operate like all other menus in dBASE III PLUS. Some submenus allow more than one option to be selected, for example, when you are building a fields list in The Assistant. When a submenu allows you to select more than one option, you indicate when you have finished making selections by pressing ← or →.

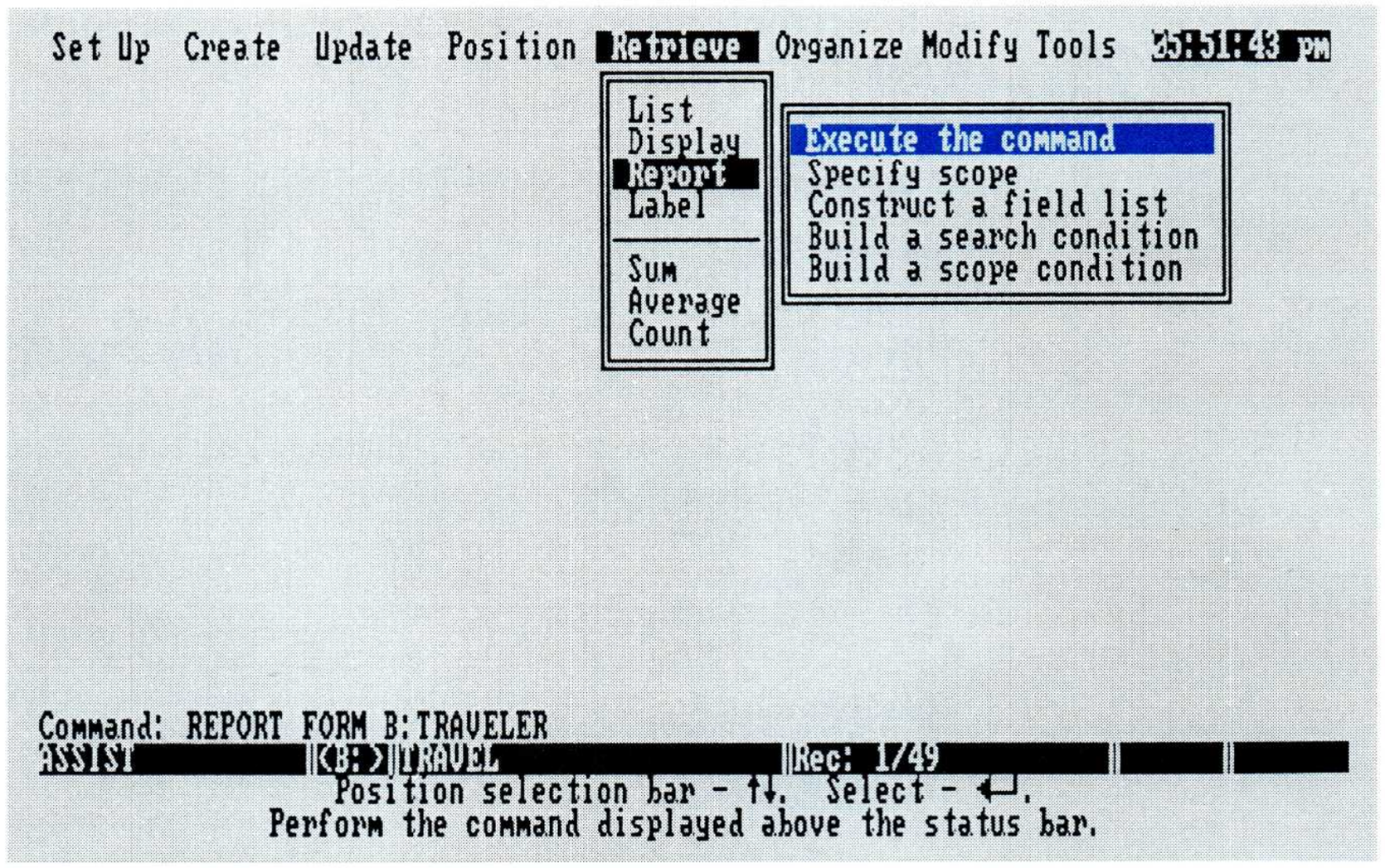



Figure 2-5 Submenu list

**User-defined Values**

Some menu selections require that you type in a response. First, you must press ↵ to open the entry area. A > appears to the left of the entry area after which you type your response. If the response can be a field name, you press **F10** and select a field from a fields submenu. Most often, you'll enter an expression in the entry area which is a combination of fields and characters that you type. When you've finished entering your response, you press ↵ again to close the entry area. Figure 2-6 displays a sample entry area.

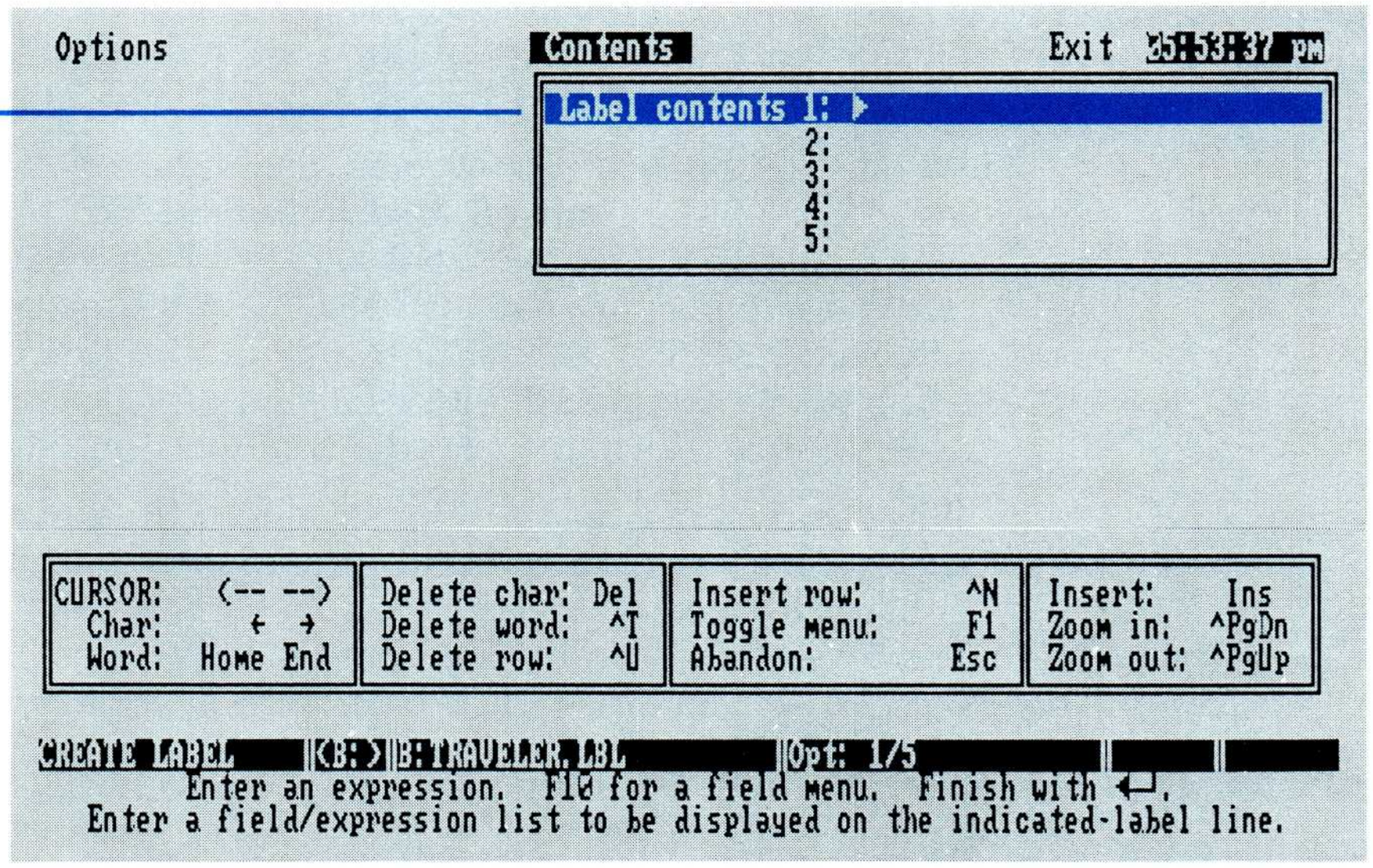



Figure 2-6 Entering a value

## Exiting Menus

To return to the dot prompt from The Assistant or the menu-assisted SET command, press **Esc**. dBASE III PLUS retains the work done with those commands. The SET options are only retained until you quit dBASE III PLUS or change them. (See Chapter 4 to find out how to save SET options.)

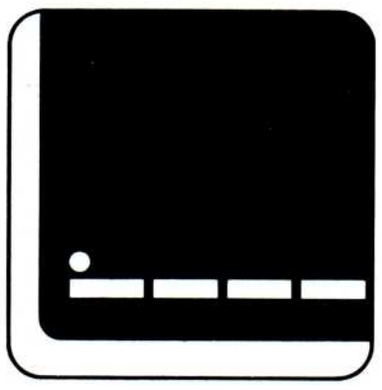
All other menu-driven commands, such as CREATE/MODIFY SCREEN, create a new file or otherwise alter the contents of a file. The Exit menu allows you to either save the work, or abandon it and retain the previous version, if one existed.

### Saving Work

To save your work, move to the Exit menu and select the **Save** option.

### Abandoning Work

To abandon the changes made or a new file created, either move to the Exit menu and select the **Abandon** option, or at any point, press **Esc**. You are asked to confirm that you want to abandon the current operation by typing Y.





## Using Screen Forms

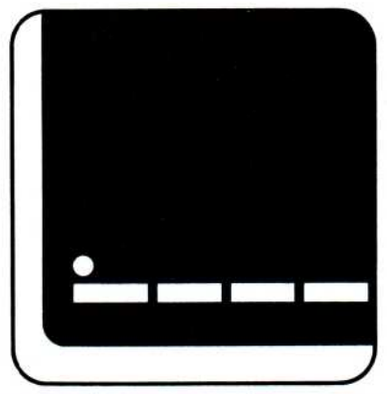
A number of commands are referred to as full-screen commands because of the way they use the screen display. These are APPEND, CREATE, EDIT, MODIFY STRUCTURE, and READ.

When you use these commands, data is presented to you as a form to be filled in or changed. CREATE and MODIFY STRUCTURE display standard forms. APPEND and EDIT also have standard forms, but you can design your own via The Assistant or the CREATE/MODIFY SCREEN command. You can also create forms using the @ command with GET and SAY options. In this case, you are building a format (.fmt) file (see Creating Screen Forms below).

When you use forms, prompts are presented in standard video (light letters on a dark background). Data items are normally presented in enhanced video or highlighted (dark letters on a light background). The arrow keys can be used to place the cursor anywhere within the highlighted areas. Data is entered by typing within these identified data areas.

**NOTE**
If you have a color card and monitor, you can change the colors of the standard and enhanced displays on your screen. You can choose colors via the SET menu and its Screen submenu or by using the SET COLOR TO command. These colors are present until you change them or QUIT dBASE III PLUS.
If you wish the colors to be present whenever you use dBASE III PLUS, save them in the Config.db file (see Chapter 4 in this manual).

For monochrome monitors, the Screen submenu has different options. You cannot choose colors, but you can change the intensity of the standard and enhanced areas of your screen and include underlining and blinking.



For example, APPEND and EDIT present each record as a standard screen form. The field names appear in standard video, and the data field areas are in enhanced video. The example below is taken from the Clients database file:

```
. EDIT RECORD 5
```

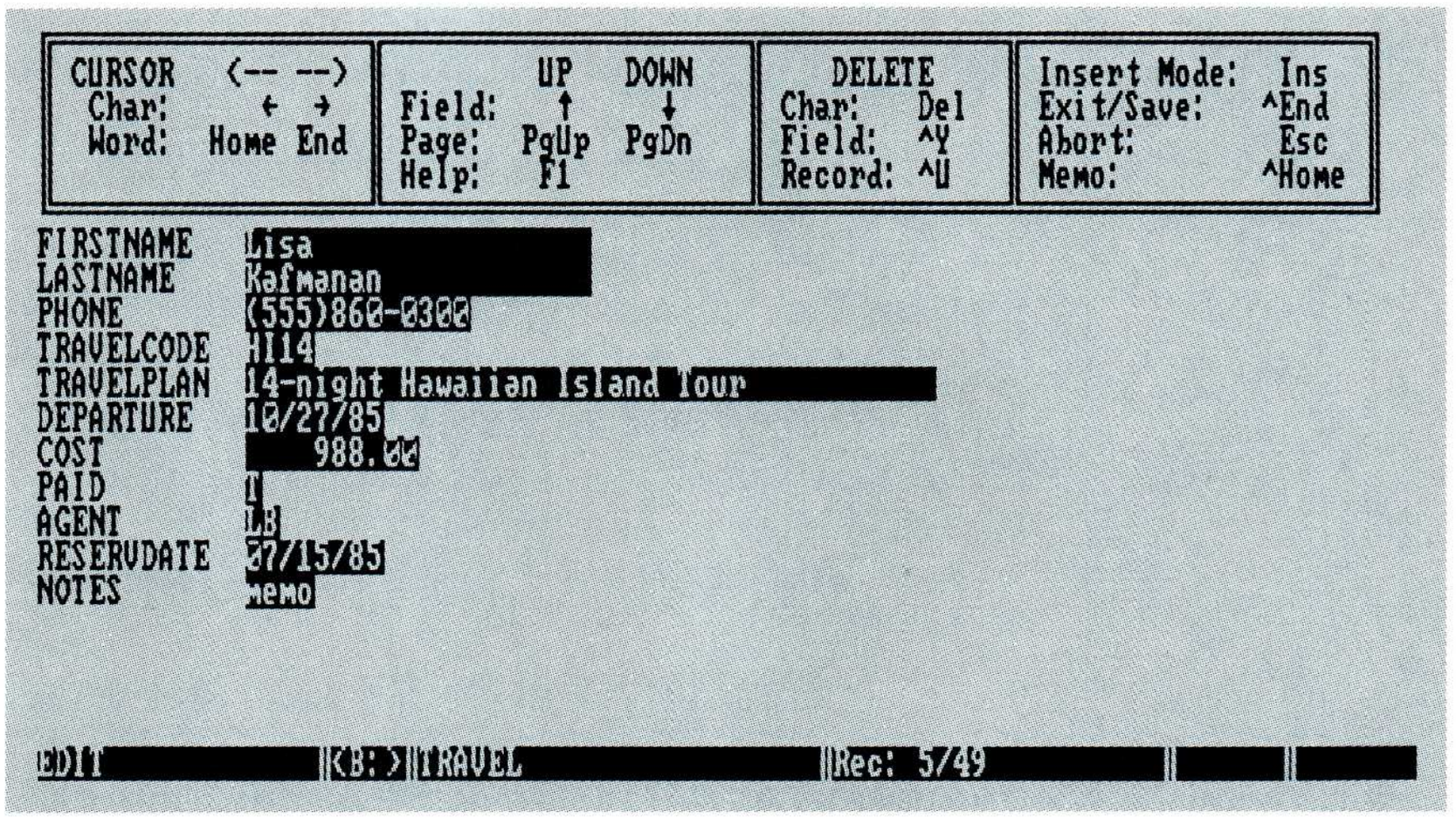
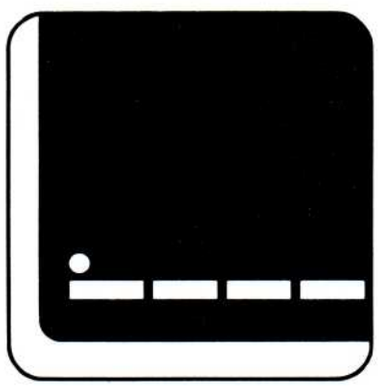



Figure 2-7 Editing records

If the database has several records, you can page backward through the records using the **PgUp** key. You page forward with the **PgDn** key. A record may be marked for deletion by using **Ctrl**-U. The word **Del** appears on the status bar for that record.

The navigation and editing keys that can be used with full-screen commands appear at the top of the screen (see Figure 2-7).

A complete description of each special key and its operation in the full-screen mode is found in Appendix A. Specific information regarding any unique command operation will be found under the particular command.





## Using Commands for Your Database

The following sections describe the commands you will use most often to manipulate your database and obtain the information you need. Read the following sections for an overview of dBASE III PLUS.

## Creating a Database File

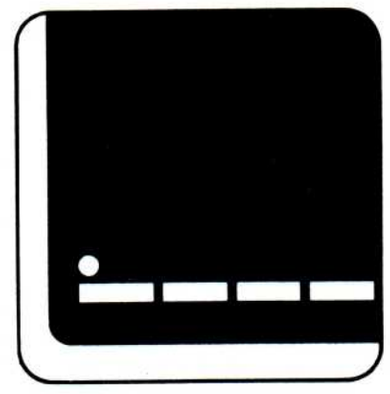
You create a database (.dbf) file by typing CREATE <filename> at the dot prompt. If you do not include a filename, dBASE III PLUS prompts for one. If you do not include an extension, .dbf is automatically appended. If you assign a different extension, you need to include it in the filename each time the file is USEd.

After entering the command, a form appears through which you define the structure of the database file. At the top of the screen is a menu of cursor and editing keys. The file structure includes the field names, types, and widths. If the field is numeric, you also specify decimal places. (See Chapter 1 for a description of fields.)

After you have defined the structure, press **Ctrl-End** to save it. dBASE III PLUS asks if you wish to **Input data records now? (Y/N)**. Answer Y and the screen form for the APPEND command appears. Answer N and the dot prompt appears for your next command.

To modify the structure of a database file, type MODIFY STRUCTURE. If you have not put a file into USE, dBASE III PLUS prompts you for a filename. The same screen form appears with the data you entered. You can change the name, type, and width of a field, or remove the decimal places in a numeric field, but you must only change one part a time.

When you save the new structure, dBASE III PLUS creates two files, a backup of the original file and a new file with the new structure. Data from the backup file is automatically copied to the new file unless you have changed field names. In this case, you are asked first if you want the backup data copied to the new file. If you do, only fields with the same name are copied.





# Adding New Records

## APPEND, BROWSE, and INSERT

You can add new records to your database file through three commands. APPEND adds records to the end of the file. It automatically updates any open index files, putting them in order. INSERT puts a record in a specified physical location in the file. Generally, you use INSERT only where it is important that the record number reflect the record's true position in the file, which is often the case with SORTed files. However, if you have index files open, INSERT works like APPEND and adds records to the end of the file, one record at a time. BROWSE has an option to add records to the end of the file.

APPEND and INSERT present the same screen form as EDIT. With both, you can SET a format (.fmt) file to custom design the screen.

SET CARRY ON copies the contents of the current record to the newly APPENDed or INSERTed record.

## APPEND BLANK and INSERT BLANK

APPEND BLANK and INSERT BLANK add a blank record to a file, but do not display a screen form for you to enter data. INSERT BLANK lets you reserve a specific place anywhere in the file. APPEND BLANK puts the blank record at the end of the file. After issuing these commands, use APPEND, BROWSE, CHANGE, or EDIT to enter data in the blank records. APPEND BLANK is often used in application programs when it is necessary to verify the accuracy of data. (See Chapter 11 in *Programming with dBASE III PLUS* for more information on this procedure.)

## APPEND FROM

The APPEND FROM command lets you copy records from a named database file to the end of the open database file. It copies only those fields that are common to both files and only those records that meet the criteria specified in the FOR statement of the command syntax.

You can use the APPEND FROM command to convert data from Visicalc, Multiplan, Lotus 1-2-3, and other programs into a dBASE III PLUS format and add it to the end of an open database file. For more information on this procedure, see the alphabetical entry for APPEND FROM in Chapter 5.

**NOTE**
As a precaution, keep backup copies of your original files, and carefully verify data in the files you've created from a file conversion.

## Changing Records

Changing the contents of a database file is one of the fundamental operations you'll perform in dBASE III PLUS. There are a variety of ways to do this, and each has unique features.

The following table outlines the commands you use to alter records in a database file.

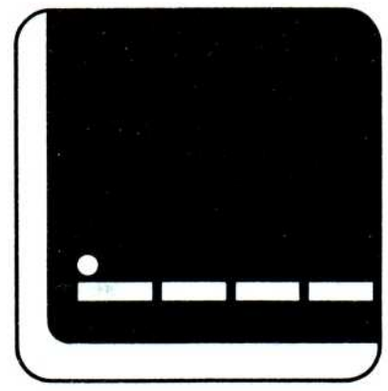
| Individual records | Batch replacements | Adding records | Deleting records |
|---|---|---|---|
| BROWSE<br>CHANGE<br>EDIT | REPLACE<br>UPDATE | APPEND<br>BROWSE<br>INSERT | DELETE + PACK<br>ZAP |
| When using APPEND, INSERT, CHANGE, EDIT, or BROWSE, you can mark records for deletion by pressing **Ctrl-U**. | | | |

Table 2-4 Changing field contents

Each command is described briefly in this section. See Chapter 5 for more information on any command.

## Changing Individual Records

If you make an error while entering data in the database file, or you want to update the contents of a particular file, you can use either the EDIT or BROWSE command. Both commands let you look only at specific fields. Below are some general guidelines for using EDIT and BROWSE.



**EDIT**

You use EDIT or CHANGE when you want to view one record at a time. (These commands are identical.) The record's appearance on the screen depends on whether you have a format (.fmt) file open. If a format file is not open, the record appears in the default vertical form. If a format file is in use, the record appears on the screen in the manner specified by the format (.fmt) file.

**BROWSE**

Use BROWSE when you want a table view of up to 17 records at once. If the records are more than 80 characters long, you can't view all of one record at a time. However, BROWSE does allow you to select which fields appear on the screen and how many characters from each character type field will be displayed.

While using BROWSE you can also add records to the end of the file and search for a key expression.

## Changing a Batch of Records

**REPLACE**

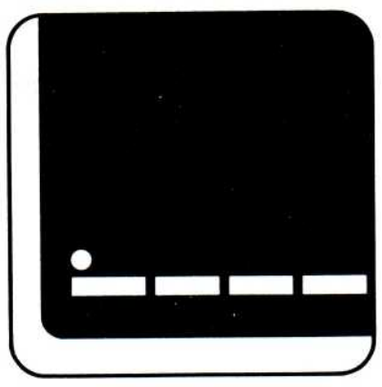
If you want to make the same change to one field throughout the file, dBASE III PLUS allows you to automate the process with the REPLACE command. For more information, see the description of REPLACE in Chapter 5.

**UPDATE**

The UPDATE command lets you replace the content of one or more fields in a database file using content from another database file. This transfer is performed when key fields in both files are the same. For more information, see the description in Chapter 5.

## Deleting Records

When you find that you no longer need a certain record, you can remove it permanently from the file. Alternatively, you can just mark it for removal. It is then available to be reinstated if you change your mind. DELETE marks records for deletion. Reinstate records that are marked for deletion with the RECALL command. The commands for permanently deleting records are PACK and ZAP. PACK removes all records marked for deletion; ZAP removes all records in the database file. These are discussed in Chapter 5.





In full-screen mode, such as EDIT and BROWSE, the word **Del** indicates that a record is marked for deletion.

In other commands, such as LIST, an asterisk (*) before the record indicates that a record is marked for deletion.

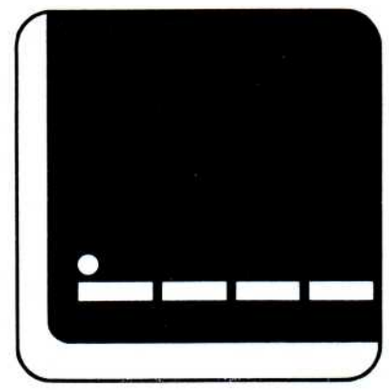
Use the **Ctrl-U** toggle when you are in full-screen mode to mark a record for deletion or to reinstate a marked record.

If you want to work with a file that has records marked for deletion, but do not want to include those records, type SET DELETED ON. Then, all dBASE III PLUS commands treat the deleted records as if they do not exist. The SET DELETED OFF command causes the marked records to be visible to dBASE III PLUS again.

## Organizing Database Files

You can enter your data in a database file in any order and then retrieve the data in a logical order: alphabetically, numerically, or chronologically. For example, you enter names in the Clients database file as you get them. However, to use the file most efficiently, the names should be in alphabetical order. Not only does this enable you to list the records in a logical order, but dBASE III PLUS can search through the file more quickly. dBASE III PLUS provides two commands that do this, INDEX and SORT.

INDEX and SORT each have advantages. With INDEX, you specify a key expression on which to order the file. A key expression can be a field name or a complex expression. SORT uses a field list and has options to establish ascending or descending order. Other differences between INDEX and SORT are summarized below.



| | INDEX | SORT |
|---|---|---|
| Creates new .dbf file | | + |
| Renumbers records | | + |
| Reorders records after EDIT/APPEND | + | |
| Orders in ascending order (A – Z) | + | + |
| Orders in descending order (Z – A) | | + |
| Orders on substrings | + | |
| Orders on complex expressions | + | |
| Orders on multiple fields | + | + |
| Orders on multiple fields of different types | +* | + |
| Can create a subset file, using scope, FIELDS, WHILE, and FOR | | + |
| Can use a file without duplicate records, with UNIQUE | + | |

* You can INDEX on multiple fields of different types, but you must form an expression by converting them to the same data type.

Table 2-5 A comparison of INDEX and SORT

For most purposes, you should use indexed files, because dBASE III PLUS automatically updates open index files. With both commands, your original file remains intact. INDEX creates an associated index file that only contains pointers to the database file from which it was created; it does not contain the actual data records. SORT, on the other hand, creates an entirely new file, which can be used independently of the file from which it was created. It is therefore up to you to maintain the order of a SORTed file.

For more details on INDEX and SORT, see the individual listings in Chapter 5.

| Record# | Lastname |
|---|---|
| 1 | Buckman |
| 2 | Lisbon |
| 3 | Bicksby |
| 4 | Garnett |
| 5 | Kafmanan |
| 6 | Johnson |
| 7 | Collins |

Entered order

| Record# | Lastname |
|---|---|
| 15 | Adams |
| 29 | Anderson |
| 9 | Arlich |
| 42 | Beman |
| 3 | Bicksby |
| 39 | Brendon |
| 1 | Buckman |

Indexed order

| Record# | Lastname |
|---|---|
| 1 | Adams |
| 2 | Anderson |
| 3 | Arlich |
| 4 | Beman |
| 5 | Bicksby |
| 6 | Brendon |
| 7 | Buckman |

Sorted order

Figure 2-8 SORT versus INDEX

## Indexing

A database (.dbf) file can have as many associated .ndx files as you want. However, only seven .ndx files can be active at a time. Using the Tours database file, you can index the file on Seats_open by typing:

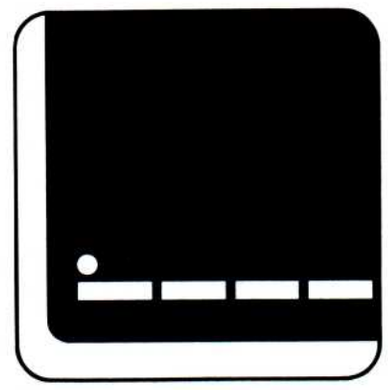
```
. USE Tours
. INDEX ON Seats_open TO Seats
  100% indexed          30 records indexed

. LIST Seats_open, Departure, Travelplan
```

If you have several existing index (.ndx) files that you want updated every time you change the contents of a record or add a new record, you must open them. There are two ways to do this:

```
. USE Tours
. SET INDEX TO <Index1>, <Index2>, <Index3>
```

or

```
. USE Tours INDEX <Index1>, <Index2>, <Index3>
```



### SET ORDER TO

In both cases, Index1 is the controlling or master index, because it is the first in the list. This means that any operation, such as LIST, acts as if the file is ordered according to the key expression of Index1. If you want to change the controlling position to another .ndx file, such as Index3, without closing and reopening the files, type:

```
. SET ORDER TO 3
```

The database file now acts in the order specified by Index3, the third file in the list.

### SET UNIQUE

There are times when you want to use a file but not include duplicate records, or you only want to include the first record with a key expression. For example, using Tours, you may want only a list of tour dates.

```
. USE Tours
. INDEX ON Departure TO Udepdate UNIQUE
  100% indexed              25 records indexed
. LIST Departure
```

Notice that there are 30 records in the database file, but only 25 are indexed. That is because records 10, 18, 19, 20, and 27 duplicate a date in a previous record.

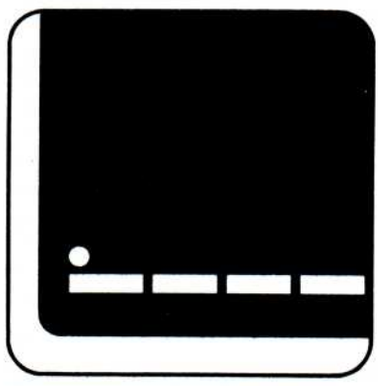
An alternative way to create a unique index file is to issue SET UNIQUE ON before INDEXing a file.

You can update UNIQUE and non-UNIQUE files at the same time, however, duplicate records in a UNIQUE file are inaccessible.

### REINDEX

Use the following commands to index files that were not open when you altered your file contents:

```
. SET INDEX TO Index8, Index9, Index10
. REINDEX
```





## Closing Index Files

There are six ways to close index files:

- SET INDEX TO ↵
- SET INDEX TO <new filenames>
- CLOSE INDEX
- QUIT
- CLEAR ALL
- Close the database (.dbf) file to which it is attached

## Sorting

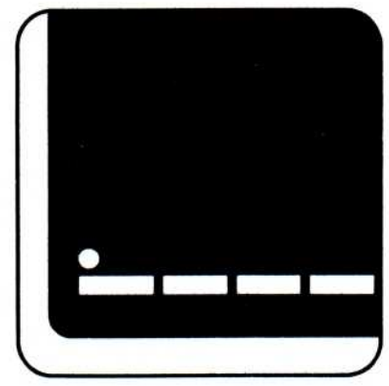
SORT is more flexible than INDEX for creating conditional files. For example:

```
. USE Clients
. SORT TO People ON Lastname/CD FOR Zipcode < '90000'
     100% Sorted   11 records sorted
. USE People
. LIST Lastname, Firstname, Zipcode
```

In this SORT command /C ignores upper case and lower case distinctions in the Lastname field. /D puts the last names in descending order, from Z to A. For Zipcode < '90000' further limits the new file to include only those records in which the zip code is less than 90000. By default, the zip codes are ordered in ascending order. This is the same as /A, for ascending.

## Searching for Data

To find specific records, such as records containing the name Ellen Michaels, use the LOCATE, FIND, or SEEK command. The appropriate command depends on whether the database file is indexed and the type of data for which you are searching.



| | LOCATE | FIND | SEEK |
|---|---|---|---|
| Index files only | | + | + |
| Works with: | | | |
| Character field | + | + | + |
| Date field | + | | + |
| Logical field | + | | |
| Numeric field | + | + | + |
| Expression | + | | + |
| Accepts scope, WHILE, and FOR in syntax* | + | | |

* Use SET FILTER TO to limit FIND and SEEK searches.

Table 2-6 Comparison of LOCATE, FIND, and SEEK

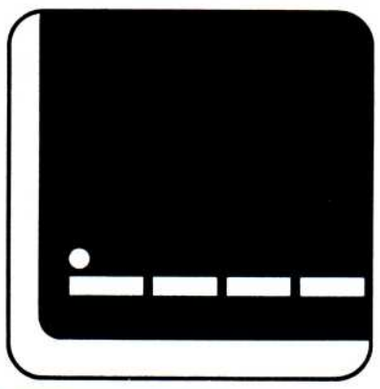
There is no way to search for specific information in Memo fields.

## LOCATE and CONTINUE

You can search indexed or unindexed files for a specific record with the LOCATE command. This moves the record pointer to the first occurrence of the string. To move to subsequent occurrences, type CONTINUE. For example, to LOCATE the tours which cost more than $2,000:

```
. USE Tours
. LOCATE FOR Unitcost > 2000
Record =   11
. DISPLAY Travelcode, Travelplan, Unitcost
Record# Travelcode Travelplan                    Unitcost
     11 KS19       19-day Kenya Safari           3195.00
. CONTINUE
Record =   13
. DISPLAY Travelcode, Travelplan, Unitcost
Record# Travelcode Travelplan                    Unitcost
     13 ME19       19-day Mediterranean          2850.00
```

When no more records meet the LOCATE/CONTINUE specifications, the message **End of LOCATE scope** appears.





## For Indexed Files: FIND and SEEK

Both the FIND and SEEK commands require an indexed file. Use FIND to move the record pointer to a record containing a specified character string or number. Do not use FIND to search for an expression or date. You must delimit the item you are looking for if it begins with leading spaces or any one of the delimiter characters: ',", [, or ].

SEEK works only with expressions, therefore, character strings must be delimited.

You also use SEEK rather than FIND in applications in which the user inputs a value to a memory variable. For instance, if a travel agent is looking for the record of a person named Michaels, the command file that accesses the database files may include an option to enter a name into a memory variable. The routine would be similar to the following from the dot prompt.

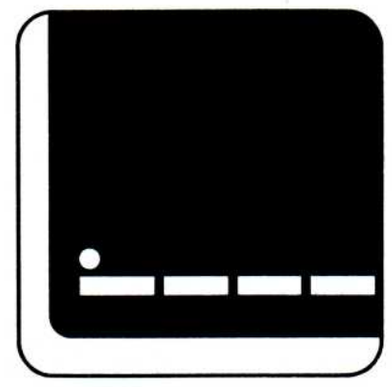
```
. mname = 'Michaels'
. USE Travel INDEX Tnames
. SEEK mname
. DISPLAY Lastname
Record# Lastname
     32 Michaels
```

To use FIND for this example, you would have to use the & function (macro function), which tells dBASE III PLUS to look for the contents of a memory variable. In the above example, you would replace the SEEK command with FIND &mname.

## Using GOTO and SKIP

You can use GOTO and SKIP to search either indexed or unindexed files. GOTO TOP and GOTO BOTTOM move the record pointer to the first and last records in the file. In an unindexed file, the first and last records are the first and last records you entered. In an indexed file, the first and last records depend on the order of the index. GOTO < n > moves the record pointer to the specified record number.

SKIP moves the record pointer forward or backward in a file. In an indexed file, SKIP follows the order of the index. In an unindexed file, SKIP moves through the records sequentially.



## Opening and Closing Files

You must explicitly open and close many dBASE III PLUS file types. These fall into two basic categories: those that make up a database and can be stored in a catalog (.cat) file, such as index and query files, and those that are text-type files, such as program (.prg) files.

Some file types are always used simultaneously. For instance, index (.ndx) files and format (.fmt) files require an open database (.dbf) file. Each file type has specific commands which open and close it.

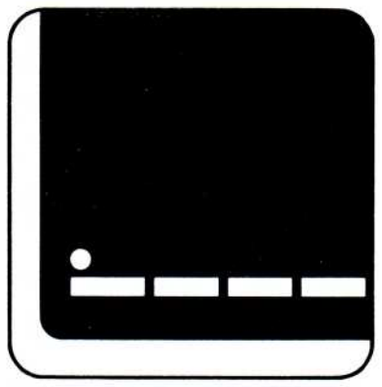
Many files close automatically when you open another file of the same type. All files close when you exit dBASE III PLUS with the QUIT command. However, it is a good practice to close files yourself when you are finished with them. This protects the data in case of a power surge or other mishap to the computer.

Sometimes when you close files, you want all associated files closed or memory variables released. With dBASE III PLUS commands, you can close either one file at a time or several.

## Number of Open Files

You can have a maximum of 15 dBASE III PLUS files open at one time. This requires you to set the DOS Config.sys FILES parameter to 20. DOS and dBASE III PLUS use five of the files, allowing you to open 15 files while running dBASE III PLUS. (See Chapter 4 of this manual for instructions on setting up and using a Config.sys file.)

If you get the message **Too many files are open**, you must close a file before opening a new one. Remember that when you open a view (.vue) file, you may be opening several database (.dbf) files and possibly some index (.ndx) files and a format (.fmt) file as well. If in doubt, use the DISPLAY STATUS command to see how many database and related files are open.





## Opening Files

Some file types are functional on their own, but others require certain other files to be open with them.

| Stand alone | Require an open .dbf file |
|---|---|
| .cat | .fmt |
| .dbf | .frm |
| .prg | .lbl |
| .vue | .ndx |
| | .qry |

Table 2-7 Relation of open files

Although .vue files can stand alone once they're built, you must have at least one open .dbf file to create them.

The following table lists the commands that open each type of file:

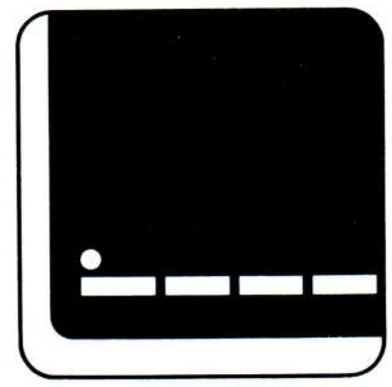
| Command | .cat | .dbf | .fmt | .frm | .lbl | .ndx | .qry | .vue |
|---|---|---|---|---|---|---|---|---|
| LABEL | | | | | + | | | |
| REPORT | | | | + | | | | |
| SET CATALOG TO | + | | | | | | | |
| SET FILTER TO* | | | | | | | + | |
| SET FORMAT TO | | | + | | | | | |
| SET INDEX TO | | | | | | + | | |
| SET VIEW TO** | | + | (+) | | | (+) | | + |
| USE | | + | | | | | | |
| USE...INDEX | | + | | | | + | | |

(+) indicates that the file type is open only if files of that type are included in the view.

* SET FILTER TO FILE <.qry filename> sets up the filter conditions stored in the .qry file. The .qry file itself does not remain open.

** SET VIEW TO <.vue filename> opens all files that make up a .vue file. The .vue file itself does not remain open.

Table 2-8 Opening files



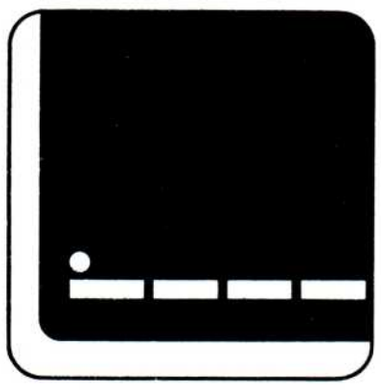
## Closing Files

Some commands create a domino effect, closing a database file and all of its open associated files. Others close only one file at a time. Some commands also release all memory variables.

| Command | .cat | .dbf | | .fmt | | .ndx | | Release memory variables |
|---|---|---|---|---|---|---|---|---|
| | | all | current only | all | current only | all | current only | |
| CLEAR ALL | + | + | | + | | + | | + |
| CLOSE ALL | + | + | | + | | + | | |
| CLOSE DATABASES | | + | | + | | + | | |
| CLOSE FORMAT | | | | | + | | | |
| CLOSE INDEX | | | | | | | + | |
| SET CATALOG TO ↵ | + | | | | | | | |
| SET FORMAT TO ↵ | | | | | + | | | |
| SET INDEX TO ↵ | | | | | | | + | |
| QUIT | + | + | | + | | + | | + |
| USE ↵ | + | | + | | | | + | |
| Several file types close automatically: .frm, .lbl, .qry, and .vue. | | | | | | | | |

Table 2-9 Closing database and associated files

With the exception of index files, only one file of each type can be open at a time in the same work area. dBASE III PLUS allows seven open index files per work area. Since a single work area can have only one database file and one format file, opening a new file of the same type closes the previous one. Although up to seven index files can be open in one work area, opening new index files closes all the open ones. Because a catalog (.cat) file is a special type of database file that resides in work area 10, you can close it either by SETting a new catalog, or by making area 10 the current work area and closing it with the USE command.





The following examples demonstrate this automatic file closing.

```
. USE Clients
. SET INDEX TO Names
. ? NDX(1)
B: Names.ndx
. SET INDEX TO Zipcode
. ? NDX(1)
B: Zipcode.ndx
```

## Using Files Simultaneously

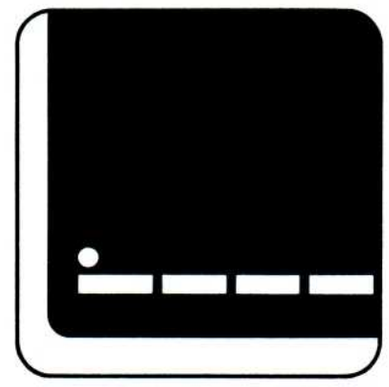
A database is a collection of related files, including database (.dbf) files and their associated index (.ndx), format (.fmt), and other files. Because dBASE III PLUS is a database management system, it allows you to use more than one file at a time.

You can work with up to ten database (.dbf) files at once. Each file must be stored in a separate work area numbered between 1 and 10. If you open a catalog (.cat) file, dBASE III PLUS automatically puts it in work area 10. This effectively lowers the number of possible open database files to nine.

> **WARNING**
> Do *not* move the catalog to a different work area. If you do, dBASE III PLUS will not be able to locate it for updates and catalog queries.

For each work area, dBASE III PLUS keeps track of:

- The database (.dbf) filename
- Any open, associated index (.ndx) files and their order
- File relations
- Any open, associated format (.fmt) file
- The SET FILTER condition
- The record pointer position



## Work Areas

Whenever you open a database file, that file is put in the currently selected work area. Unless you have issued the SELECT command to activate a different work area, work area 1 is active. To open a second database file without closing the first one, use the SELECT command to move to another work area. When you move between work areas, the files you open in a work area remain open until you close them.

If only one file is in USE, you do not have to concern yourself with work areas.

Study the following example:

```
. USE Tours
. SELECT 2
. USE Clients
. GO 3
. DISPLAY
Record#  Firstname  Lastname  Address         City       State  Zipcode  Phone
      3  Hank       Bicksby   4101 Peonia Rd  Flagstaff  Az     86001    (555) 966-1278
. SELECT 1
. GO 5
. DISPLAY
Record# Travelcode Travelplan          Departure Unitcost Percent Seats_open  Note
      5 CN7        7-night Cancun Trip 10/07/85  759.00   7.500   87          Memo
. SELECT 2
. DISPLAY
Record#  Firstname  Lastname  Address         City       State  Zipcode  Phone
      3  Hank       Bicksby   4101 Peonia Rd  Flagstaff  Az     86001    (555) 966-1278
. USE
. CLOSE DATABASES
```

Notice that USE closed only the files in work area 2. CLOSE DATABASES closed all database files and their associated files.

When you have several database files open, you can transfer data from one file to the other. You can also position the record pointers in several files simultaneously. See the Linking Files with SET RELATION section below for more information.

## Aliases

Internally, dBASE III PLUS keeps track of a database file both by its filename and by its ALIAS. You can assign an ALIAS, or dBASE III PLUS can use the filename as an ALIAS by default. If you use an ALIAS name, you can SELECT a work area by its ALIAS name, rather than by its number. Choosing an alias name that is descriptive, or using the default filename, is easier to remember than a number.

If you want to assign an ALIAS, do it when you open the file:

```
. USE Travel ALIAS Trip
. SELECT 2
. USE Clients
. DISPLAY STATUS

Select area:  1, Database in Use:  B:Travel.dbf  Alias: TRIP
          Memo file:  B:Travel.dbt

Currently Selected Database:
Select area:  2, Database in Use:  B:Clients.dbf  Alias: CLIENTS
```

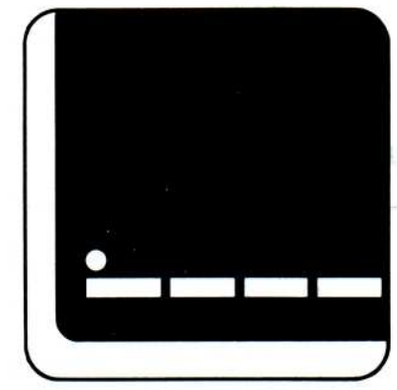
Here's an example of working with two files. If Travel.dbf is the currently SELECTed file, and you want to find out the name of the person in the current record in Clients.dbf, type:

```
. ? Clients->Lastname
Buckman
```

The alias symbol (->) is used to indicate that the field name belongs to the preceding ALIAS. The file must be in USE with the defined ALIAS in an unSELECTed work area. Note that the alias symbol is entered with two keystrokes, a hyphen (-) and a greater than symbol (>).

## Linking Files with SET RELATION

When creating a data management system, you often put static information, such as client names, addresses, and telephone numbers, in one file (Clients.dbf) and information subject to change, such as bookings for a travel package, in another file (Travel.dbf). Often, there are fields with the same information, such as Lastname and Firstname in Clients.dbf and Travel.dbf.



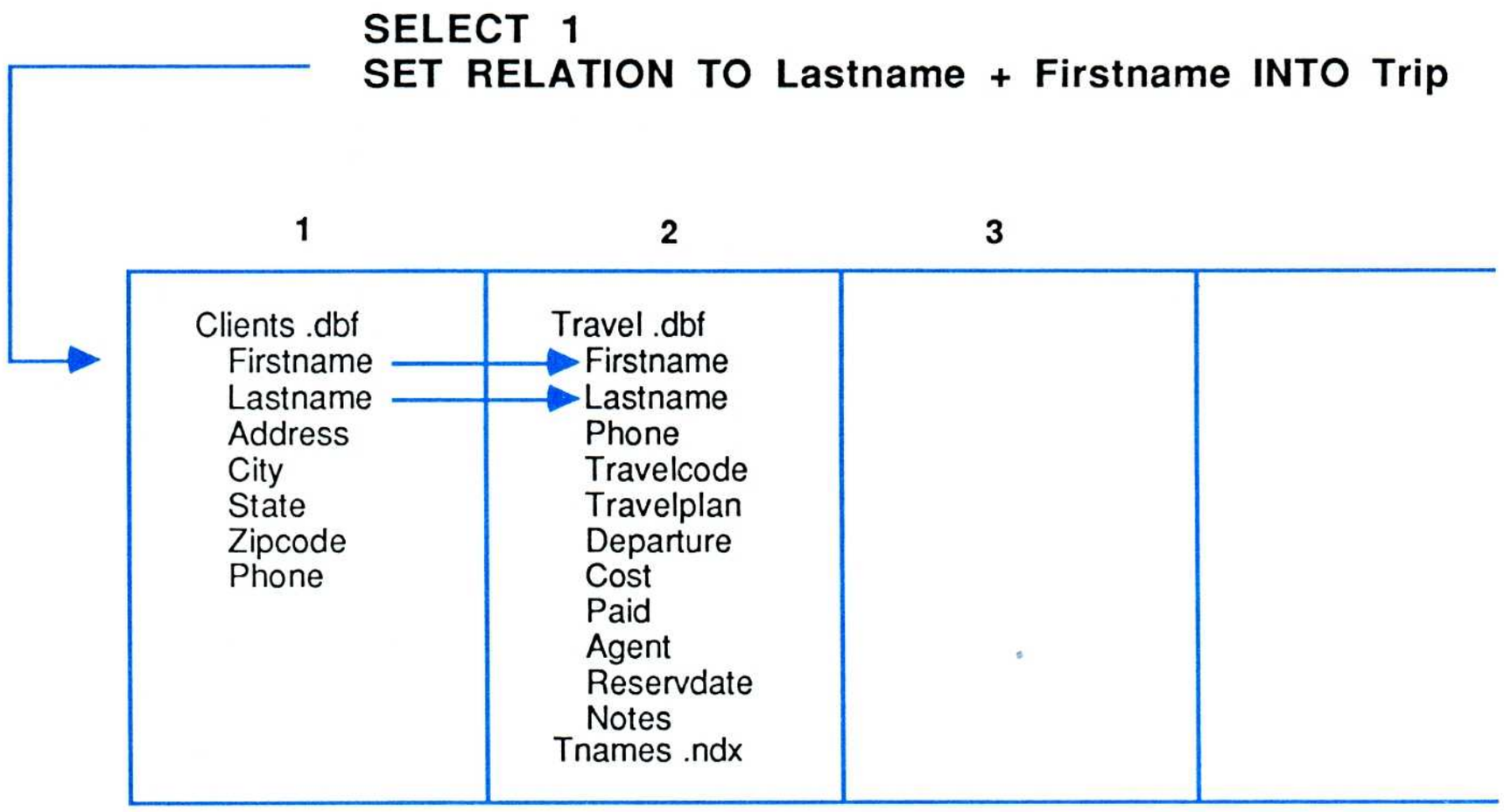
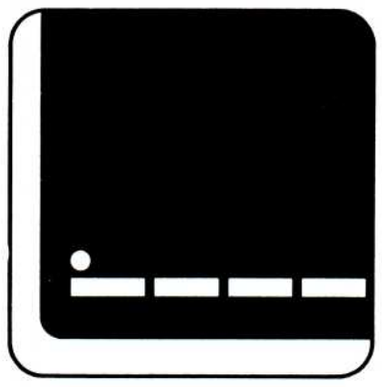



Figure 2-9 Relating database files

To use these files as one file, you must link them with a common field, expression, or record number. This common link is the index key in the related database file. Between Clients.dbf and Travel.dbf, the link is the expression **lastname + firstname** as shown in Figure 2-9.

With such a link, you can correlate names and addresses with bookings, repeating only a minimum of information in both files. For example, if you want to send travel information to everyone who booked a trip in August, you would enter the following commands to extract their names, addresses, trip name, and date of departure from the database:





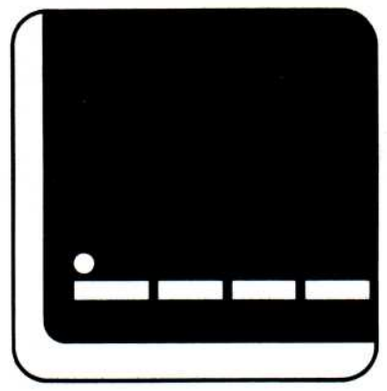
```
. SELECT 1
. USE Clients
. SELECT 2
. USE Travel ALIAS Trip
. SET INDEX TO Tnames
. SELECT 1
. SET RELATION TO Lastname + Firstname INTO Trip
. SET FILTER TO Trip-> Departure > CTOD('07/31/85') .AND.
Trip-> Departure < CTOD('09/01/85')
. DISPLAY ALL Lastname, Firstname, Trip-> Departure
Record# Lastname      Firstname          Trip-> Departure
      2 Lisbonn       Rick               08/05/85
     13 Youngblood    Dick               08/23/85
     19 Daniels       Dominique          08/15/85
     26 Johnson       Mike               08/23/88
     31 Rizzo         Ann                08/15/85
     33 Peters        John               08/23/85
     41 Larson        Jill               08/23/85
```

Having set up the index file based on Lastname + Firstname, the SET RELATION TO command causes the record pointer to move in both files, not just one. When the record pointer moves to a new name in the SELECTed file, Clients, it also finds the first occurrence of that name in the Trip file.

To unlink files, you type SET RELATION TO ↵.

If you often correlate the same files in the same way, you can establish the relation and store it in a file for repeated use. The file containing this information is called a view (.vue) file which you create, modify, and use with these commands: CREATE VIEW FROM ENVIRONMENT, CREATE/MODIFY VIEW and SET VIEW TO.

With this information stored in a view (.vue) file, you can use the CREATE LABEL command to make mailing labels for all the people on selected trips; summarize the data with CREATE REPORT; or just LIST the names, telephone number, and departure dates for a particular trip.



## Storing a SET RELATION to a View (.vue) File

If you've been working from the dot prompt but want to save your setup, first check that everything is set up correctly:

- The SET RELATION is right.
- All necessary database and index files are open.
- SET FILTER TO is in effect, if it is being used.
- A format file is open, if one is to be used.

If those conditions are present, you can save this working environment for future access in a view (.vue) file.

To save the environment, you type: CREATE VIEW <filename> FROM ENVIRONMENT. dBASE III PLUS supplies the .vue extension.

The next time you need this setup, type SET VIEW TO <filename> at the dot prompt.

See CREATE VIEW FROM ENVIRONMENT and SET VIEW TO in Chapter 5 for more information.

## Creating and Modifying a View File

The preceding sections showed you how to set up a relationship between two files, create a filter condition for the linked files, and store the entire environment in a view (.vue) file.

Another way to accomplish the same tasks is to use the CREATE VIEW command. When you type CREATE VIEW <filename> at the dot prompt, a menu appears from which you can select database files and fields and set up a relation as if you were at the dot prompt. You can also set up a filter condition and format file, and then save the view.

Any view file created using CREATE VIEW FROM ENVIRONMENT or CREATE VIEW can be modified using MODIFY VIEW.

To use a view file, you type SET VIEW TO <filename> at the dot prompt. dBASE III PLUS activates the view and the dot prompt reappears. To deactivate a view file, open another view file, or type CLOSE DATABASES to close all files in the view.

For more information on creating or modifying a .vue file, see the CREATE/MODIFY VIEW entry in Chapter 5.

CHAPTER 2

# Displaying and Printing Data

dBASE III PLUS offers three methods of turning raw data into a more presentable form. You can:

- Create reports and labels
- Design custom screen forms
- Send control codes to your printer for special printing effects

## Reports and Labels

In dBASE III PLUS, you can create reports and labels from your database files. The commands you use are CREATE/MODIFY REPORT and CREATE/MODIFY LABEL. Each of these commands is menu-driven, using the standard menu interface for dBASE III PLUS.

## CREATE/ MODIFY REPORT

With CREATE/MODIFY REPORT, you can design reports in a columnar layout, using selected fields from from either a database (.dbf) file or a view (.vue) file. You can also include expressions, memory variables, and, if necessary, functions to convert data types. CREATE/MODIFY REPORT enables you to design a structure for the report. You do not see the actual data in a report until you print it out.

There are several options for designing a page, including page titles, headings, and page numbers. You can divide your data into groups or subgroups, each with its own heading, or create a summary report only. Numeric fields can also be totalled.

While you choose options to design the report, you will also see a diagram of your report showing the headings and column widths you've defined for the selected fields. This diagram can help you "see" what your report looks like as you design it. Finally, you can save the report structure in a report form (.frm) file for future use.

To print the report, you use the REPORT command and its options from the dot prompt. For more information, see the CREATE/MODIFY REPORT or REPORT commands in Chapter 5 of this manual. For a step-by-step example of creating a report, see Chapter 6 in *Learning dBASE III PLUS*.

## CREATE/ MODIFY LABEL

CREATE/MODIFY LABEL enables you to design standard mailing labels, using selected fields from database files and view files. With these commands, you design a label form that you save in a label (.lbl) file. You do not see actual data in the label form until you print it out.

The menu offers five standard label sizes but you may also define different sizes. You define the label contents by specifying fields, memory variables, or expressions. If necessary, you can also use functions to convert data types. Before exiting the menu, save the file for future use.

From the dot prompt, use the LABEL command to print out labels. For more information, see the CREATE/MODIFY LABEL and LABEL commands in Chapter 5. For a step-by-step example of creating labels, see Chapter 6 in *Learning dBASE III PLUS*

## Creating Screen Forms

For many purposes, the format in which dBASE III PLUS presents records is sufficient. For example, if you EDIT or APPEND records, each record appears on the screen in the default screen form.

However, when you write a custom application for another user, you may want to set up a more visually appealing screen, or try to match a paper form on which to print the data. You can create the form yourself and store it as a *format (.fmt) file*. The Clients.dbf file structure could appear on the screen like this:

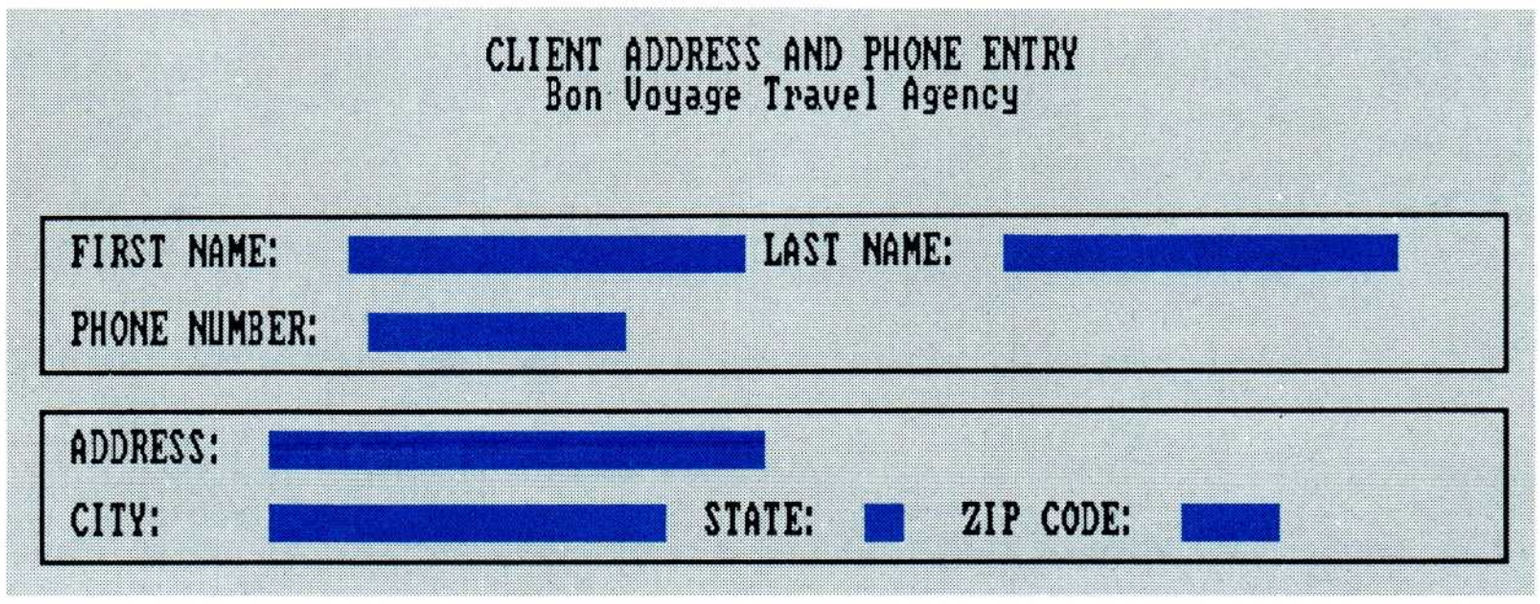
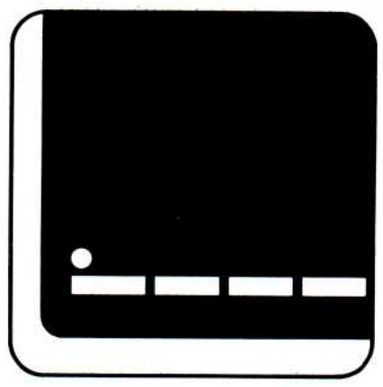



Figure 2-10 Custom screen form





Format (.fmt) files consist of a series of @ commands that tell dBASE III PLUS where to position each of the fields and how to introduce each one. One advantage of using format files is that you don't have to use the field names to identify the fields. You can use more descriptive phrases to indicate what information to put in the field. For example, you can identify the Departure field with **Departure Date**.

Another advantage of using format (.fmt) files is that you have more control over data. You can restrict what type of data is acceptable, such as characters only, or specify a range of dates or numbers like 1 through 15. To keep the data consistent, you can specify that data will be converted to upper case regardless of how it is entered. For more details on designing format (.fmt) files, see the **@** command entry in Chapter 5 of this manual.

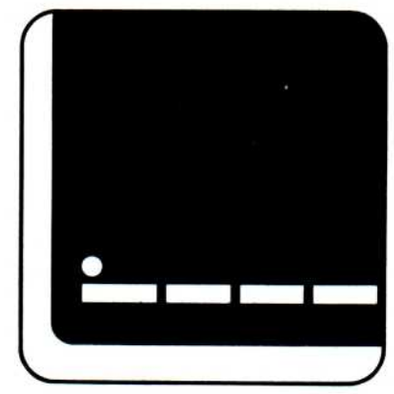
Custom screens are also useful while working with several related files. Before creating the format (.fmt) file, link the files with either SET RELATION TO or CREATE/MODIFY VIEW. If you link the files with CREATE/MODIFY VIEW, you can also add a format (.fmt) file that contains fields from the linked files. In this case, whenever you SET VIEW TO <filename>, the associated format (.fmt) file opens automatically.

There are two ways to create a format (.fmt) file: with CREATE/MODIFY SCREEN, the dBASE III PLUS Screen Painter, or with a text editor such as MODIFY COMMAND, to specify the @ commands yourself. Each is discussed in Chapter 5, "Commands."

## CREATE/ MODIFY SCREEN

When you type CREATE SCREEN or MODIFY SCREEN at the dot prompt, you are prompted for the screen filename, then the Screen Painter menu appears.

If you have not already done so, your first step is to select a database file. Next you set up the fields to be included in the form. After that, you open the *blackboard* and start placing the fields. On the blackboard, you can also change the field names to make them more descriptive. If you are working with only one file (and not several related ones), you can permanently add, delete, or otherwise change fields in the file structure. To make menu selections, press **F10** to move between the blackboard and the menu bar.



When you save the screen form, dBASE III PLUS stores the information in a screen (.scr) file and automatically creates a matching format (.fmt) file containing @...SAY...GET statements. You can modify the screen file via MODIFY SCREEN but you can't work with it directly. The screen file is, in essence, only a tool for creating a format (.fmt) file. Each time you alter and save the screen file, the existing format (.fmt) file is automatically updated.

**NOTE**

- Do *not* use double quotes in the field descriptions as they will generate the error message **Unrecognized phrase/keyword in command** when you use the format file.
- Do *not* ERASE the screen (.scr) file if you want to modify the format (.fmt) file later. Once the screen file is gone, it can't be re-created from the associated format file. If you ERASE the screen, you can only make changes to the format file directly, by changing the @...SAY...GET command statements.
- Do *not* change the format file with MODIFY COMMAND or another text editor if you intend to modify it again with MODIFY SCREEN. The Screen Painter reads the screen file only, which won't include the format file changes. When you save the screen file, it overwrites the previous format file, which eradicates any changes made to it by means other than the Screen Painter.

## The @ Command

Another way to create a format file is to determine the row and column coordinates for each field. For instance, the format file for the screen form in Figure 2-10 looks like this:

```
. use clients
. type clients.fmt
@  4, 21  SAY "CLIENTS ADDRESS AND PHONE ENTRY"
@  5, 24  SAY "Bon Voyage Travel Agency"
@  9,  1  SAY "FIRST NAME:"
@  9, 14  GET  CLIENTS->FIRSTNAME
@  9, 37  SAY "LAST NAME:"
@  9, 50  GET  CLIENTS->LASTNAME
@ 11,  1  SAY "PHONE NUMBER:"
@ 11, 16  GET  CLIENTS->PHONE
@ 15,  1  SAY "ADDRESS:"
@ 15, 13  GET  CLIENTS->ADDRESS
@ 17,  1  SAY "CITY:"
@ 17, 13  GET  CLIENTS->CITY
@ 17, 38  SAY "STATE:"
@ 17, 46  GET  CLIENTS->STATE
@ 17, 52  SAY "ZIP CODE:"
@ 17, 63  GET  CLIENTS->ZIPCODE
@  8,  0  TO 12, 79
@ 14,  0  TO 18, 79

:
Command Line  |B: >|CLIENTS  |Rec: 1/49  |  |
Enter a dBASE III PLUS command.
```

Figure 2-11 A format (.fmt) file

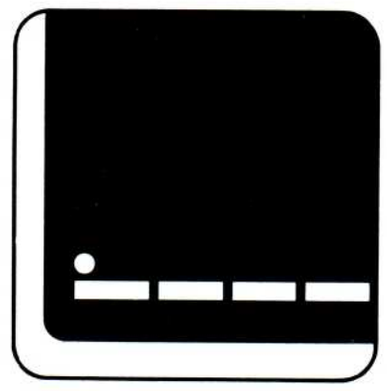
With this method, you can use the dBASE III PLUS text editor or any word processor that creates standard ASCII text files.

## Opening Format Files

There are two ways to activate a format (.fmt) file:

1. You can USE a database (.dbf) file and then SET FORMAT TO <filename>.
2. If while setting up a view (.vue) file you chose a format (.fmt) file that references selected fields in the view, the format file is automatically opened whenever you SET VIEW TO <filename>.

After opening the format file, the APPEND, CHANGE, EDIT, INSERT, or READ commands use the screen form.



## Closing Format Files

To close a format file, type either CLOSE FORMAT or SET FORMAT TO ↵.

When you close a database (.dbf) file in a work area, any open format (.fmt) file in the area is automatically closed.

## Printing Format Files

To print the form without any field contents, type SET DEVICE TO PRINT. This command sends only @...SAY statements to the printer. @...GET statements are ignored.

To print data using a custom screen form, you need to modify the format file and change all GETs to SAYs. To do this, you should use the MODIFY SCREEN command and change all fields from EDIT/GET to DISPLAY/SAY. This ensures that both the screen and format files match and reflect the changes. To get a printout, first type SET DEVICE TO PRINT then DO <format filename> including the .fmt extension

## Controlling Your Printer

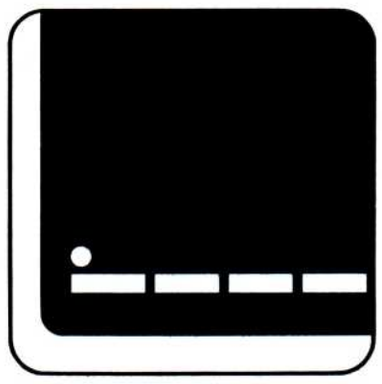
There are two ways to print data in dBASE III PLUS:

- Using the dBASE III PLUS printing commands
- Sending escape code sequences to your printer

If your printer supports graphics characters, or a variety of type styles and sizes, you can control them in dBASE III PLUS. You just need to know your printer's control codes. Using control codes, you can also set up your printer for different form lengths and to print headers and footers.

The dBASE III PLUS printing commands work with all printers. Both the commands and printer escape code sequences can be issued from the dot prompt or included in program files.

For more information on sending control codes, see the description of the CHR() function in Chapter 6 of this manual or Chapter 12 in *Programming with dBASE III PLUS*. Table 2-10 lists the dBASE III PLUS printing commands.

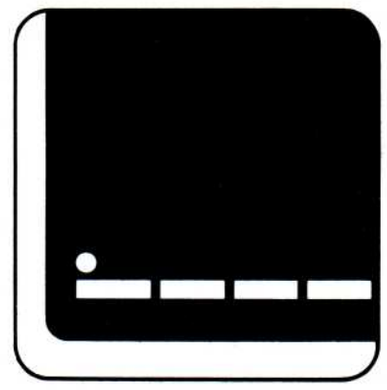




| | |
|---|---|
| ?/?? < expression > | Sends an expression or its result to the printer, when SET PRINT is ON |
| DISPLAY/LIST TO PRINT | Directs database output to the printer |
| TYPE < filename > TO PRINT | Directs specified file to the printer |
| EJECT | Causes the printer to advance one sheet of paper |
| LABEL FORM TO PRINT | Directs a label (.lbl) file to the printer |
| REPORT FORM TO PRINT | Directs a report (.frm) file to the printer |
| SET DEVICE TO PRINT | Sends the results of @...SAY commands to the printer |
| SET MARGIN TO < n > | Sets the printer's left margin |
| SET PRINT ON/OFF | Activates and deactivates the printer |
| SET PRINTER TO < DOS device > | Directs the output to a DOS printer other than the default < LPT1 > , for example, COM1 or LPT2 |

Table 2-10 dBASE III PLUS commands for printing.

For more information on these commands, see the individual entries in Chapter 5 of this manual.



## The Data Catalog

Data catalogs (.cat files) provide a way to organize database files, view files, and all files that require a database file to be in USE for their creation. These include:

| File Type | Extension |
|---|---|
| format | .fmt |
| index | .ndx |
| label | .lbl |
| query | .qry |
| report form | .frm |
| screen | .scr |

Table 2-11 Files that require an open .dbf file

A catalog is similar to a database file, except that it has a .cat extension rather than a .dbf extension. Like a database file, it has a structure, fields, and records. Each record describes one file in the associated group of files. Catalog files keep track of the same information for all groups of associated files, regardless of application, so the structure of every catalog is the same and has the same field names.

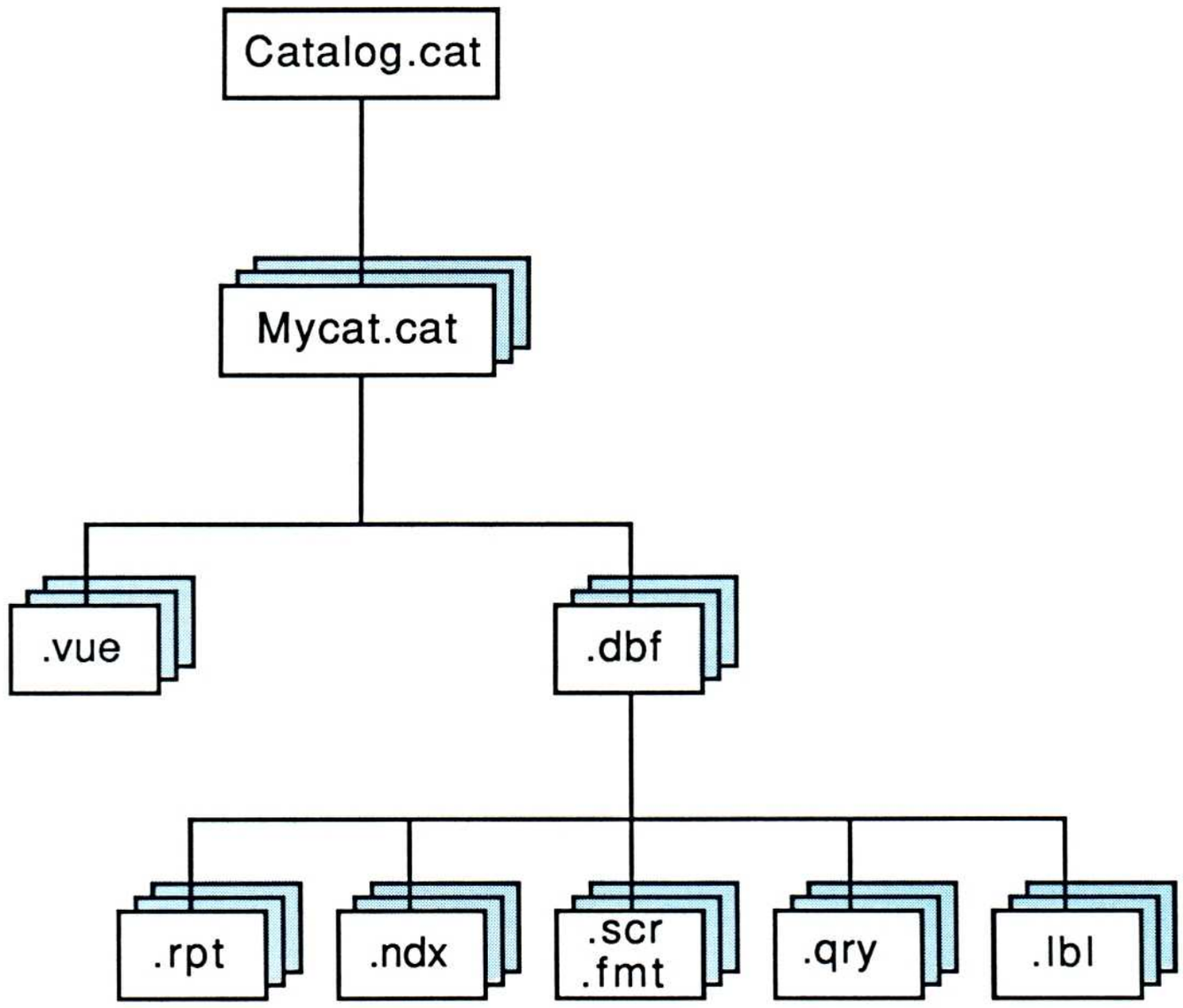
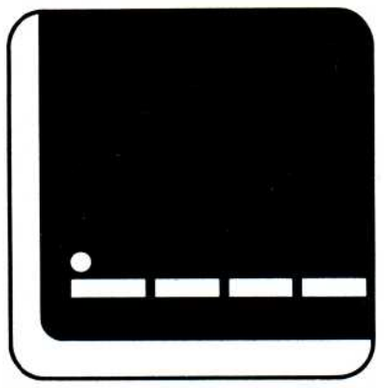



Figure 2-12 Contents of a catalog file

Catalogs keep track of which auxiliary files belong with which database file. You register a group of files into a catalog when the files have a connection with one another. For instance, suppose you have a set of files that relate to your accounts and another set for a project that includes bibliographies and research notes. All files relating to accounts can go in the catalog Accounts.cat and all research files in Research.cat.

Then, whenever you open one of those catalogs, dBASE III PLUS acts as if only the files in that catalog exist. For example, when you issue the Catalog query clause with the command USE ?, a menu of database (.dbf) files in the open catalog appears.



## Creating a Catalog

You create a new catalog to house a database by entering SET CATALOG TO <filename> at the dot prompt. dBASE III PLUS prompts you to add a title that defines the contents of the Catalog by displaying **Enter File title.** This title can be up to 80 characters long. dBASE III PLUS adds a .cat extension to the filename and adds the filename to the master catalog, Catalog.cat.

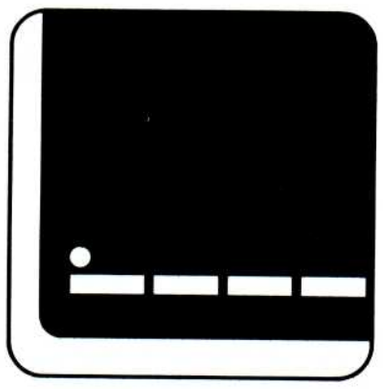
dBASE III PLUS automatically creates Catalog.cat the first time you create a catalog. Catalog.cat is stored in the default directory or subdirectory. Its purpose is to help you keep track of the catalogs you set up in the directory and to let you select one from a menu of catalog names.

> **NOTE**
> Avoid naming your catalogs Catalog.cat since that name is reserved for the master catalog.

## Catalog Structure

All catalogs have the same file structure. With the exception of the Title and Tag fields, dBASE III PLUS automatically fills in the field contents whenever a new file is entered in the catalog. The structure is:

| *Field* | *Field Name* | *Type* | *Width* | *Dec* |
|---|---|---|---|---|
| 1 | PATH | Character | 70 | |
| 2 | FILE_NAME | Character | 12 | |
| 3 | ALIAS | Character | 8 | |
| 4 | TYPE | Character | 3 | |
| 5 | TITLE | Character | 80 | |
| 6 | CODE | Numeric | 3 | 0 |
| 7 | TAG | Character | 4 | |
| ** Total ** | | | 181 | |





There are instances in which a field is intentionally left blank. For example, because only database files have ALIAS names, the ALIAS field is only used to define database (.dbf) files. For more information on catalog structure, see the description of SET CATALOG in Chapter 5.

## Opening a Catalog

To open an existing catalog, type SET CATALOG TO <filename> or SET CATALOG TO ? to display a menu of all catalogs. Whenever a catalog is opened, work area 10 is reserved for it.

SET CATALOG TO <filename> opens the named catalog. However, if you misspell the filename or enter the name of a nonexistent catalog, dBASE III PLUS assumes that you want to create a new catalog.

SET CATALOG TO ? opens a menu of all catalog names in the master catalog. You then use the arrow keys and ↵ to select a catalog.

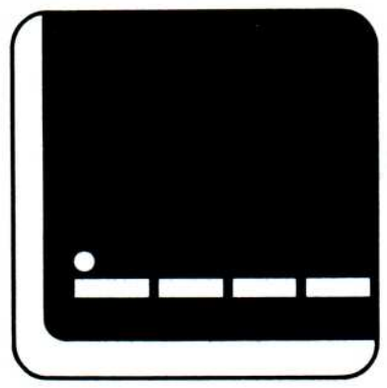
At any time in a dBASE III PLUS session, you can change the active catalog with the SET CATALOG TO command. If you want to close a catalog, type SET CATALOG TO ↵ or open a different catalog.

For more information, see SET CATALOG TO in the alphabetical listing of commands in Chapter 5.

## Adding Entries

A new entry is added automatically to the active catalog when you use any of the following commands:

COPY STRUCTURE
COPY STRUCTURE EXTENDED
COPY TO
CREATE
CREATE FROM
CREATE/MODIFY LABEL
CREATE/MODIFY QUERY
CREATE/MODIFY REPORT
CREATE/MODIFY SCREEN
CREATE/MODIFY VIEW
IMPORT FROM
INDEX
JOIN
SET FILTER TO FILE
SET FORMAT
SET VIEW
SORT
TOTAL
USE



If the filename already exists but was not in the catalog, you are prompted for a file title. Note that if you have an active catalog and then open a view (.vue) file, the database files USEd in the view file will not be added to the catalog.

**NOTE**
Use the SET TITLE OFF command to suppress the file title prompt, if you do not want to enter file titles. To prevent the catalog from being updated without closing the catalog, use the SET CATALOG OFF command.

## Deleting Entries

DELETE FILE and ERASE automatically remove a filename from the active catalog.

## Renaming Entries

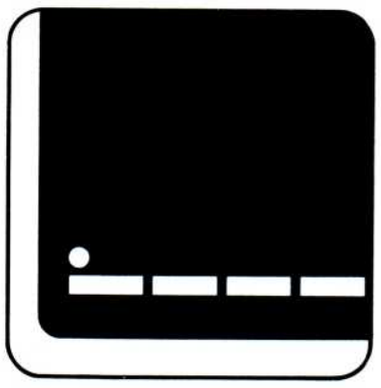
RENAME automatically changes the name of a file in the active catalog.

**WARNING**
Do not try to modify the structure of a catalog file. If you do, dBASE III PLUS will not be able to use or maintain the catalog properly.

## Closing Catalogs

The following commands close, or deactivate, the open catalog:

CLEAR ALL
CLOSE ALL
QUIT
SET CATALOG TO ↵





## The Catalog Query (?) Clause

Once a catalog is open, the catalog query clause (?) becomes another option in the syntax of certain commands. Most commands that use a filename as part of their syntax can use the catalog query instead of a filename, for example, USE ? or MODIFY REPORT ?. The file type, of course, must have already been recorded in the catalog.

The ? displays a list of filenames in the catalog. You then move the selection bar to the appropriate filename and press ↵ to activate the file.

For example, to open a menu list of all index (.ndx) files associated with Clients.dbf, type:

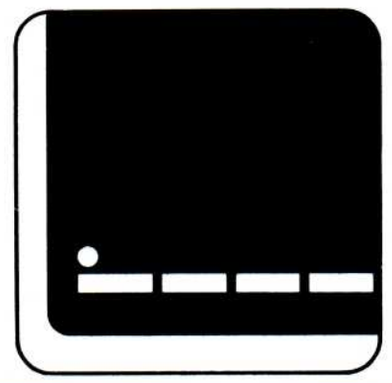
```
. SET CATALOG TO Triplog
. USE CLIENTS
. SET INDEX TO ?
```

The commands in which catalog queries apply are:

APPEND FROM
CREATE/MODIFY LABEL
CREATE/MODIFY QUERY
CREATE/MODIFY REPORT
CREATE/MODIFY SCREEN
CREATE/MODIFY VIEW
DELETE FILE
ERASE
LABEL FORM
REPORT FORM
SET CATALOG
SET FILTER TO FILE
SET FORMAT
SET INDEX
SET VIEW
USE

If you use the catalog query clause when no catalog is open, dBASE III PLUS displays the message **Data Catalog has not been established.**

You can use the catalog query command SET CATALOG TO ? to open a catalog or enter the SET CATALOG TO command followed by the name of a catalog.



## Special Features of dBASE III PLUS

Several commands let you modify one or more existing files to create a new file. You can:

- Make exact copies of any type of file (COPY FILE)
- Copy all or only selected portions of a file (COPY)
- Create a file that contains the sums of numeric fields (TOTAL)
- Merge information from two files to create a third (JOIN)

You can also use files from other programs through the APPEND FROM, COPY, IMPORT, and EXPORT commands.

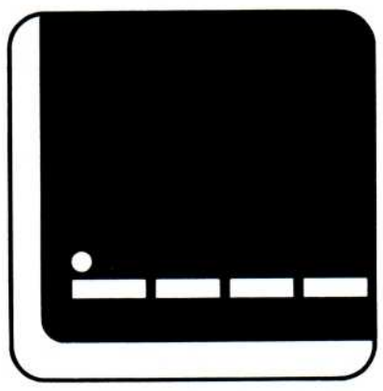
## Using Other Programs

dBASE III PLUS has a built-in converter for importing files from other programs to dBASE III PLUS files, and exporting dBASE III PLUS files to other programs. These include:

| File | Type |
|---|---|
| Lotus 1-2-3 Spreadsheet | WKS |
| Multiplan Spreadsheet | SYLK |
| pfs:File | PFS |
| VisiCalc | DIF |

Table 2-12 Importing and exporting files.

**WARNING**
As a precaution, always keep backup copies of all original data files, and verify that the data in the files you've created through a file conversion is both complete and accurate.





In addition to these specific programs, you can use other files that are in standard ASCII format, such as MailMerge™ files in WordStar®. The two ASCII formats that dBASE III PLUS accepts are:

| ASCII format | Type | Description |
|---|---|---|
| Delimited | DELIMITED | Commas or blanks separate fields. Delimiters, such as double quotations, enclose character fields. A carriage return and line feed separate records. |
| System Data Format | SDF | Fields are of fixed length and a carriage return and line feed separate records. |

Table 2-13 ASCII formats.

DELIMITED files include MailMerge and BASIC sequential access files. SDF files include data files created by a FORTRAN program.

To use Framework with dBASE III PLUS database (.dbf) files, use the converter in Framework.

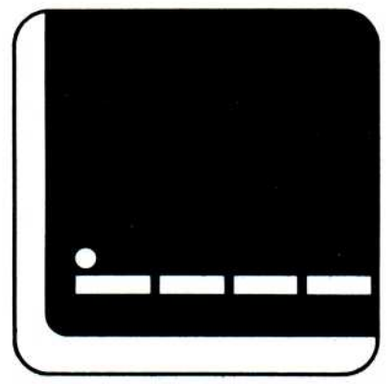
## Importing Files

There are two commands for importing file, IMPORT and APPEND FROM.

The IMPORT command converts a file from pfs:File to a dBASE III PLUS database file and format file. There is a 4,000-byte record limit on the size of database files you can create with dBASE III PLUS.

If a catalog is open, the files are entered in it. A view (.vue) file is created as well as a database (.dbf) and a format (.fmt) file. When you activate the view using SET VIEW TO <filename>, the screen appears as it did in pfs:File.

APPEND FROM has more available options than IMPORT, but it does not have the capability to create new files.



If the FROM file is a dBASE III PLUS database file, its records are added to the end of the open database file.

If the FROM file is *not* a dBASE III PLUS .dbf file, you must include the TYPE of file from the types (except PFS) listed in Tables 2-12 and 2-13.

## Exporting Files

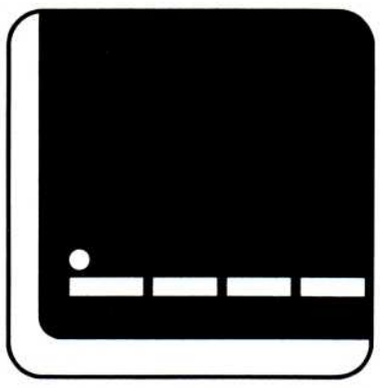
There are two commands for exporting dBASE III PLUS files to other programs, EXPORT and COPY. They follow the same principles as IMPORT and APPEND FROM.

The EXPORT command converts a database (.dbf) file and an associated open format (.fmt) file to a PFS file. If a format (.fmt) file is not open, the dBASE III PLUS default file structure becomes the pfs:File screen format.

The COPY command has more options than the EXPORT command, however, it cannot merge a database (.dbf) file with its associated format (.fmt) file. With COPY, you can create duplicates of all of the file types (except PFS) listed in Tables 2-12 and 2-13. You can also COPY subsets by including a scope, WHILE, and/or FOR clause in the command.

## Merging Data from Two Files

The main power of a database is that you can work with several files at one time. You can also take specific fields and records from two of these files and create a third file by using the JOIN command. For more information, see the alphabetical listing in Chapter 5.





## Calculating Totals

Using the TOTAL command, you can calculate totals from numeric fields in database files and store the information in a new database file. For example, to summarize the work of the salespeople at a travel agency, you could create a new file as follows:

```
. USE Travel
. INDEX ON Agent TO Reps
100% Indexed              49 records indexed
. TOTAL ON Agent TO Cost
     49 Records (s) totalled
      4 Records generated
. USE Cost
. LIST Agent, Cost
Record #    Agent         Cost
       1    JT         24863.00
       2    LB          9650.00
       3    MM         19705.00
       4    RA         21808.00
```

The one-line TOTAL command:

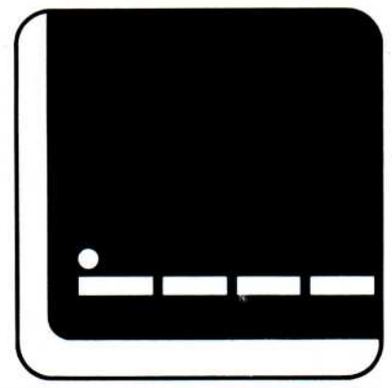
1. Creates Cost.dbf with the same structure as Travel; Cost.dbf will *not* contain any memo fields.
2. Examines each record to determine the agent.
3. Totals the cost of all trips the agent handled.
4. Creates one record for each agent.

## Copying Database Files

The COPY command is versatile and very important. With it, you can:

- Make backup copies of a file to a different drive.
- Create a file that contains selected records, determined by the scope, WHILE, and FOR options.
- Create a file that contains selected fields, determined by the FIELDS option.
- Make a copy of the file structure, without any records.

COPY requires that a database file be in USE. If the database file has an open index (.ndx) file, the new file will be in indexed order.



If the database file has records marked for deletion and SET DELETED is ON, only the unmarked records are copied to the new file. If SET DELETED is OFF, all records are copied, and the marked records retain their deleted status in the new file.

## Copying Any Type of File (COPY FILE)

COPY FILE allows you to copy any type of file, including database (.dbf), command (.prg,), and non-dBASE III PLUS files. The file must be *closed*.

This command copies files in 512-byte blocks only. Therefore, a .dbf file copied in this manner may be larger than the original file. For instance, if the original file is 600 bytes, the new file will be 1,024 bytes. COPY FILE operates faster than the COPY TO command because it copies the file in blocks.

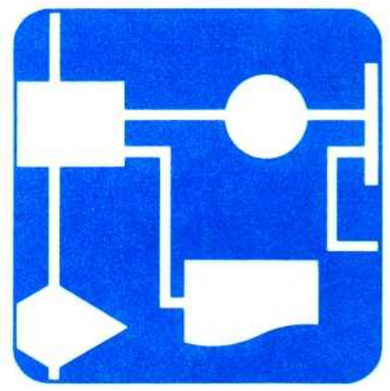
## Protecting Your Data

All files should be backed up (that is, copied) on a regular, systematic basis to protect against loss. This may be accomplished either from within dBASE III PLUS or from the operating system.

To back up from within dBASE III PLUS, use the COPY or COPY FILE command.

Any type of file may be copied this way as long as it is not open or in USE. Associated database (.dbf) and database memo (.dbt) files must be copied separately when using this method. Wildcard characters such as * and ? are not permitted in the COPY FILE command.

# Chapter 3
## ABOUT PROGRAM FILES



Running dBASE III PLUS from the dot prompt was discussed in Chapter 2. You can also run dBASE III PLUS by means of program (.prg) files in which the instructions you enter at the dot prompt are combined. Any command or function you enter at the dot prompt can be used in a program file. In fact, the command or function is written exactly the same way in a program file. In addition, dBASE has commands that you use only in program files and that give you more control over your data.

This chapter tells you about the structure of a program file, and the dBASE commands you need to create, alter, execute, close, and debug it.

This chapter tells you:

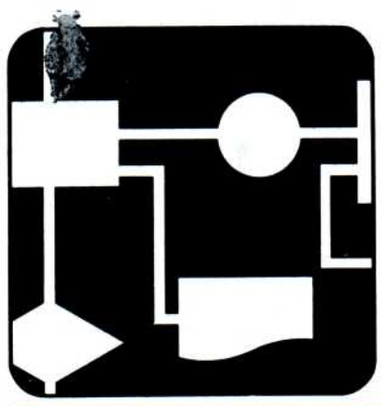
- Why you should use program files
- About special commands for program files
- The structure of a program file
- How memvars are used in program files
- How to create, execute, close, and debug program files
- Some advanced features of using program files

This chapter does not teach you how to program or design and write an applications program in dBASE; these topics are covered in *Programming with dBASE III PLUS* in Volume Two. If you need to know the syntax or use of a specific command or function, refer to Chapters 5 and 6 in this manual.

### Why Do I Need Program Files?

Using program files to run dBASE III PLUS can save you time. For example, if you need to submit a report on overdue accounts every week, you could store the commands to create the report in a program file called Overacct.prg. Then when you need the report, you only have to type DO OVERACCT at the dot prompt.

You can also use a program file to complete a more sophisticated task like designing a menu. You can even write several program files which do all your tasks, except typing in your own data. In this case, the program files are usually referred to as a dBASE applications program.



CHAPTER 3

Once you've written an applications program, you might want to market it. If you're interested in this, read *RunTime +* in Volume Two. This manual describes two programs, dBCODE and dBLINKER, which encrypt your program files and make them run faster.

## Programming Commands

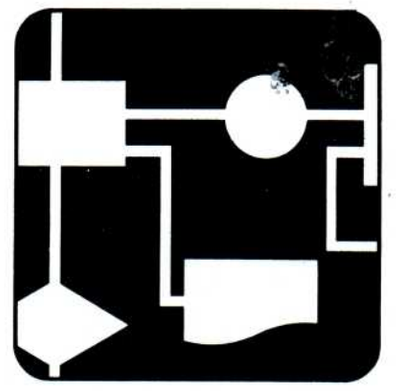
dBASE III PLUS has commands that provide data handling methods similar to those in most programming languages. These commands make programming in dBASE possible and enable program files to do more than store a set of commands you enter at the dot prompt. These commands are:

- DO WHILE...ENDDO
- IF...[ELSE]...ENDIF
- DO CASE...ENDCASE
- the ON command

These commands are described below. They have in common the ability to initiate a series of commands depending on the existence of a specified condition. Notice that the first three commands, unlike other dBASE commands, require an ending statement. It is important to remember the ending statement, since that's how dBASE III PLUS knows to execute the next command.

**TIP**
The IIF() function, a shortcut for the IF...[ELSE]...ENDIF construction, and the ON command can also be used from the dot prompt.

DO WHILE...ENDDO enables you to execute a series of commands repeatedly as long as a specified condition is true. The program repeats the same commands until the condition is no longer true.



You use IF...[ELSE]...ENDIF to specify a condition which may or may not be present. dBASE III PLUS evaluates the condition and, if it's present, executes the commands given. If the condition is not present, dBASE III PLUS performs the commands following ELSE (if included), or continues to the next command following the ENDIF statement.

In either of these commands, you only specify one condition. However, in a DO WHILE construction, dBASE III PLUS checks the condition repeatedly until it is false. In an IF...ENDIF construction, dBASE III PLUS evaluates the condition only once.

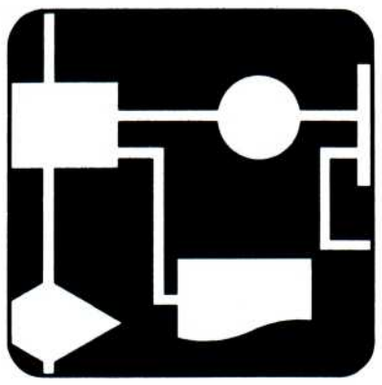
Often, you may have several possible conditions you wish to evaluate. DO CASE...ENDCASE allows you to list the different conditions and what to do with each one.

For more complex processing, you may use any of the mathematical, relational, logical, and string operators to combine expressions within a condition. (See Chapter 2 for a discussion of operators.)

You can also put any of the three commands within another, which is called *nesting*. When nesting, the statements must fit entirely within one another. If, for example, you nested an IF...ENDIF statement within a DO WHILE construction, both the IF and its accompanying ENDIF must be contained between the DO WHILE and the ENDO. Do not nest the commands too deeply, since that will make it more difficult for you to document and keep track of the work being done.

Normally, commands in a program file are executed line-by-line in the order written. DO WHILE...ENDDO, IF...ENDIF, DO CASE...ENDCASE also allow you to branch to another program file and *run it without returning to the dot prompt*. The program file you are branching from is the calling program; the program file you are branching to is the subprogram.

To execute a subprogram, you specify DO <program filename>, usually after a condition. When the condition exists, control is transferred from the calling program to the subprogram. The subprogram instructions are executed before returning control to the calling program, where the remaining instructions are processed. It is a good practice to include RETURN as the last command in a subprogram.





The ON command permits branching to a subprogram when a dBASE III PLUS error occurs, or the **Esc** key or another key is pressed. The command has three forms which test for these conditions: ON ERROR, ON ESCAPE, and ON KEY. A sample use is ON ERROR DO < program filename >. The ON command is normally placed in the setup area of a program file (the setup area is explained below) and stays active until a new ON command is issued.

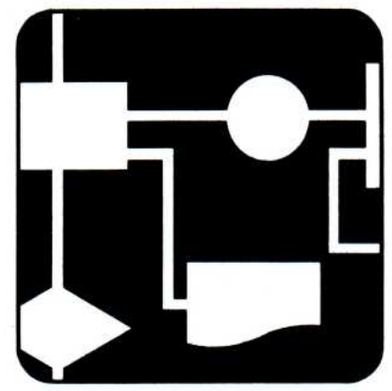
While learning these four commands is important, since they make programming in dBASE possible, program design is equally important. Good program design requires both practice and patience. How one designs a program varies widely depending on the programmer and the application. For more information on using these commands and program design, read *Programming with dBASE III PLUS*.

## Structure of a Program File

In dBASE III PLUS a program file has four basic parts (also see Chapter 2 in *Programming with dBASE III PLUS*):

- A preamble
- A setup area
- The body of the program
- A closing section

The preamble includes the program filename, a description of what it does, who wrote it, and dates indicating when it was started or last edited. Each line with this information must be preceded by the command NOTE or an asterisk (*), and followed by a ↵. Using NOTE or the asterisk tells dBASE III PLUS to ignore the information because it is not a dBASE command or program instruction. This feature is especially useful when you have several program files to keep track of, you are writing an applications program, or someone else needs to know how your program file works.



## ABOUT PROGRAM FILES

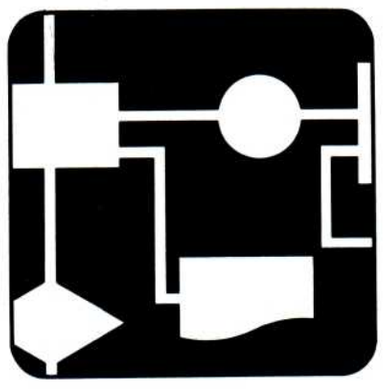
The setup area contains commands which design the operating environment for the program file. There are some aspects of the dBASE III PLUS environment that you must change for a program file; these are established by the SET command and its options. For example, you should use SET TALK OFF to turn off the interactive mode, which is the default method for running dBASE III PLUS from the dot prompt. That way, the response to certain dBASE commands will not interfere with the program flow. If you are writing an applications program, then you may want to program the function keys, SET ESCAPE OFF so your users cannot press **Esc** to stop the processing of your program, or use SET COLOR TO for special effects on the screen.

The body of the program file contains the commands which do the work of the program. In this section, you open files, select work areas, change or display database information, perform calculations, produce reports and so on — using the same commands you would enter at the dot prompt. See the Classes of Commands section in Chapter 5 for an overview and brief description of all the dBASE commands.

Also in this section are the dBASE commands used specifically for program files, which give you more control over data. These commands are summarized in the Programming Commands section above, and discussed thoroughly in *Programming with dBASE III PLUS*.

You should also document the tasks accomplished in the program. You may add comments on separate lines, beginning each line with NOTE or an asterisk (*) as described. You may also put a comment on the same line as a command by preceding the comment with two ampersands (&&).

The closing section contains commands which tidy up the work space used by your program file. For example, you would return the environment to its default setup, remove memory variables, close all files, and return to the dot prompt. (See Chapter 13 in *Programming with dBASE III PLUS*.)



CHAPTER 3

## Memvars and Program Files

You can use any type of memory variable (character, numeric, date, or logical) in a program file. They are usually initialized with STORE < value > TO < memvar name > or < memvar name > = < value >; however, you can also use the ACCEPT, AVERAGE, COUNT, PARAMETERS, INPUT, SUM, and WAIT commands to initialize memory variables. You use memory variables to store (in memory) a value to be processed or evaluated in the program file.

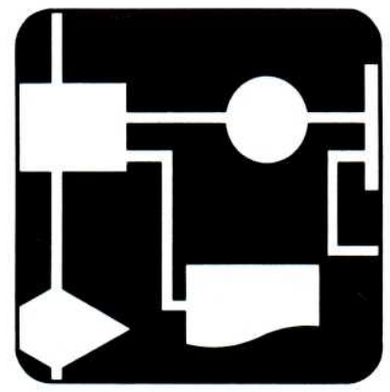
You can have up to 256 active memory variables with a combined size of up to 6,000 bytes. If your computer has more than 256K, you can increase the 6,000-byte capacity by setting the MVARSIZ option in Config.db. The contents of memory variables are only stored temporarily until you change them, release them, or QUIT dBASE III PLUS. However, you can SAVE memory variables to a memory (.mem) file, then RESTORE them later.

Memory variables initialized or RESTOREd at the dot prompt always have a *public* status. Their values are available to any operations you perform at the dot prompt, including the execution of a program file. However, memory variables initialized or RESTOREd within a program file have a *private* status by default. Their values are only accessible within the program file and to any subprograms. You can change or hide the status of a variable via the PUBLIC and PRIVATE commands.

Memory variables are normally released in the closing section of a program file.

## Functions and Program Files

You use functions in program files the same way you use them from the dot prompt (see the Using Functions section in Chapter 2). Although all dBASE III PLUS functions can be used in program files, two particularly useful functions for program files are INKEY() and READKEY(), which evaluate data input. The INKEY() function allows you to execute certain commands or branch to another program when a certain key is pressed. For example, you could use INKEY() to indicate that when the letter E is pressed, the program branches to an Edit.prg subprogram.



You use the READKEY() function to determine what action was taken to leave a full-screen operation such as APPEND, EDIT, or BROWSE. For example, you can determine whether the user pressed **Esc** or **Ctrl**-W to exit BROWSE. If **Ctrl**-W was pressed, you could have the program branch to another subprogram. If **Esc** was pressed, you could have the program return to the main menu.

See Chapter 6 for more information on the keys you can use with these functions and the actions they perform.

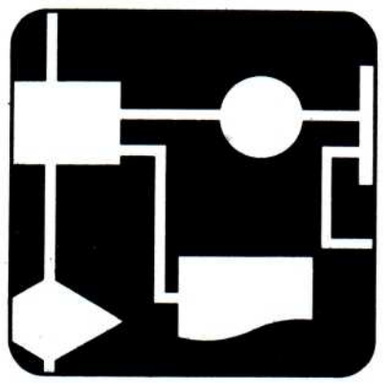
## Creating Program Files

You create program files by typing MODIFY COMMAND < filename > at the dot prompt. The filename may be up to eight characters long and have an extension. If you do not assign an extension, dBASE III PLUS automatically appends .prg.

Typing this command automatically brings up the dBASE III PLUS text editor. The filename and menu of text editing commands appear at the top of the screen. You may toggle this menu ON or OFF with the **F1** key.

Type in the instructions you wish to have executed, then save the file with **Ctrl**-W.

If you wish to modify the program file, you also type MODIFY COMMAND < filename > at the dot prompt, then resave the file with **Ctrl**-W.





**NOTE**
MODIFY COMMAND runs the dBASE III PLUS text editor. It is primarily used to create and edit program files, but you can use it when you need to create or modify a standard ASCII text file. For example, you can use MODIFY COMMAND to create format (.fmt) files (see Creating Screen Forms in Chapter 2) or to incorporate the commands into a program file.

You can also use a word processor that creates standard ASCII text files to write your program files. To run another word processor with MODIFY COMMAND, you need to include the TEDIT command in Config.db. See Chapter 4 in this manual and read the sections on Config.db for more details.

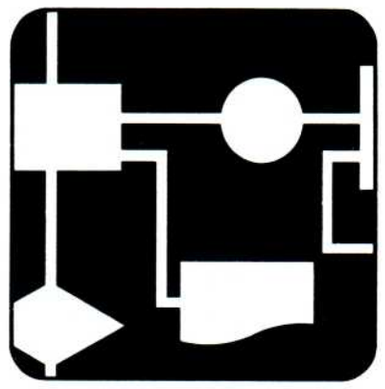
## Executing Program Files

You enter DO <program filename> at the dot prompt to open and execute a program file. dBASE III PLUS interprets and then executes the instructions line-by-line until the end of the program file. If the program file branches to a subprogram, dBASE III PLUS executes the subprogram, then returns to the calling file to execute the remaining instructions.

You can also execute a program file when you load dBASE III PLUS. At the DOS prompt, you would type dbase <program filename>. To automatically execute a program file after starting dBASE III PLUS, you need to specify DO <filename> after the COMMAND option in Config.db. See Chapter 4 for details on using Config.db.

## Closing Program Files

When dBASE III PLUS runs a program file, it executes the instructions line-by-line in the order given. When a program file ends, it is automatically closed. Table 3-1 shows the commands which close program files.





| Command | Files Closed | | Release Private Memvars | Returns Control To |
|---|---|---|---|---|
| | Current | All | | |
| CANCEL | | + | + | Dot prompt |
| RETRY | + | | | Calling program |
| RETURN | + | | + | Calling program |
| RETURN TO MASTER* | | + | + | Highest level calling program |
| QUIT | | + | + | Operating system |

* RETURN TO MASTER closes all command files except the original calling .prg file.

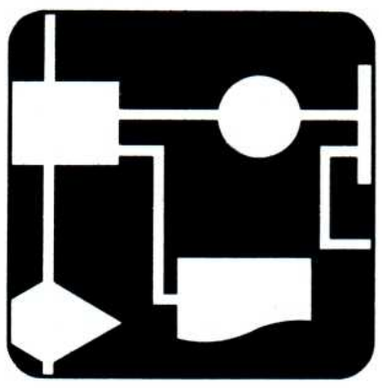
Table 3-1 Closing program files

## Debugging Program Files

Mistakes are often made when creating or writing a program file. You might have syntax errors such as a forgotten comma, space, or quotation mark, or misspelled commands and memvar names. The expressions in the program may have mismatched data types. You can also have run time errors which are not found until you actually run the program. Most run time errors are mistakes in logic. You may think your instructions are correct, but the program does not do what you expect.

There are several dBASE commands that enable you to test and debug your program files. They are RESUME, SET DEBUG, SET DOHISTORY, SET ECHO, SET HISTORY TO, SET STEP, and SUSPEND. The commands you use depend on how you like to work.

One popular method is to use SET STEP ON, SET ECHO ON, and SET DEBUG ON. With SET STEP ON, program files are executed one instruction at a time. Program execution is stopped after each command and you receive a prompt to continue, suspend or cancel execution. SET ECHO ON displays the program instructions on your screen. SET DEBUG ON prints the output from SET ECHO. Note that if you also SET TALK ON, you can see the results of some of the commands you entered (such as STORE or SUM).





Another method is to SET DOHISTORY ON to store the commands that were executed from your program file. The commands are stored in the history buffer. Only the last 20 executed commands are stored, unless you increase the size of the buffer with SET HISTORY TO.

If dBASE III PLUS finds an error, it stops executing the program, displays an error message, and asks you if you wish to ignore the error or cancel or suspend the program. With this method, press S to suspend the program and display the dot prompt. Next you type DISPLAY HISTORY to find and edit the incorrect instruction. Then you resume program execution by typing RESUME. Editing an instruction in the history buffer does not automatically update the program file. You must keep track of the changes you make and edit the program file yourself later.

For a full discussion of debugging methods, read Chapter 15 in *Programming with dBASE III PLUS*. For more details on a specific debugging command, go to Chapter 5 in this manual.

## Some Advanced Features

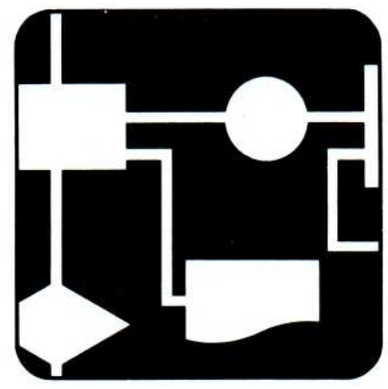
This section describes some programming techniques to make your program files more efficient.

### Procedure Files

The program file you will probably create most often with MODIFY COMMAND is also called a *command file*. Command files do the bulk of the work in a program or an application.

dBASE III PLUS has another type of program file called a *procedure file*. A procedure file contains utility programs (also known as procedures) that perform simple tasks you do very often in dBASE III PLUS. For example, if you need to print a lot of documents and like to use different fonts, you could create procedures which send the different fonts to the printer.

Unlike command files, a procedure file is stored in memory (if you have sufficient memory to store it). This means that dBASE III PLUS does not need to access the disk drive to run procedures as often. Instead, they are executed much more quickly from memory.



Like command files, you create procedure files by entering MODIFY COMMAND < filename > at the dot prompt. A procedure file is also assigned a .prg extension, unless you assign a different one.

You can have up to 32 procedures in the procedure file. Each procedure must begin with PROCEDURE < procedure name > and as a good programming practice, it should end with a RETURN statement.

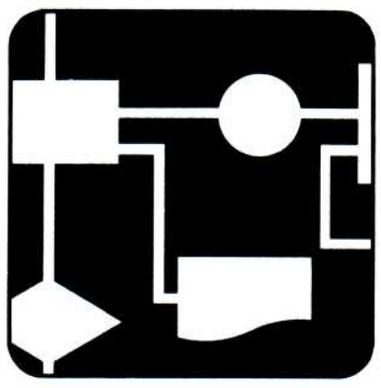
To execute a procedure file, you specify SET PROCEDURE TO < filename > at the beginning of a command file to store the procedure file in memory. When you need a procedure in the file, you use DO < procedure name > to run it. dBASE III PLUS branches to the procedure and executes it. The RETURN statement at the end of the procedure returns control to the calling command file or to the dot prompt.

You close a procedure file by specifying CLOSE PROCEDURE or SET PROCEDURE TO in the closing section of the calling command file. Because you can only have one procedure file in memory, SET PROCEDURE TO < filename > automatically closes the current one and loads the named file into memory. Note that a procedure file stays open until specifically closed, even after the calling program has been executed.

## Other Features

Another feature of program files is the ability to hide PUBLIC variables. For example, if you have a PUBLIC variable called Mprice with a value of 100, you can go to a subprogram, create a new PRIVATE variable with the same name, and give it a different value. dBASE III PLUS uses the new value only while running the subprogram and its branching files. When dBASE III PLUS returns to the controlling program, Mprice is reinstated as a PUBLIC variable with a value of 100.

Another advantage is that you can branch to binary (.bin) files. These are written in assembly language and give you more control over the hardware of your computer, for example, aspects which control cursor movement. LOAD stores a binary file in memory, where it can be executed via the CALL command.





For details on procedure files, binary files, hiding PUBLIC variables, and techniques for creating more efficient program files, read Chapter 16 in *Programming with dBASE III PLUS*.

# Chapter 4

## dBASE III PLUS CONFIGURATION

dBASE III PLUS can be customized with two configuration files: Config.sys and Config.db. These files contain commands that perform special functions and enhance your use of dBASE III PLUS.

When you receive your dBASE III PLUS package, the Config.sys and Config.db files are included on System Disk #1 and are ready for you to use. This chapter explains why these files are important and how you can modify them to tailor dBASE III PLUS to meet your needs.

### Config.sys

Config.sys is a DOS file that may contain five DOS configuration commands: BREAK, BUFFERS, DEVICE, FILES, and SHELL. Only the BUFFERS and FILES commands are needed to run dBASE III PLUS properly. Without them, dBASE III PLUS could not perform some of its data handling tasks: opening several files at one time, setting relations between database files, using view files or catalogs... In short, you would not have the power and data handling capabilities dBASE III PLUS gives you.

FILES refers to the number of files that can be open at any time. BUFFERS designates the number of buffers allocated in memory to read/write information to/from a disk.

The remaining commands are not needed for dBASE III PLUS, however, if you have other software that require these commands, their presence in Config.sys will not affect the performance of dBASE III PLUS.

The Config.sys file on System Disk #1 contains the following information:

```
FILES = 20
BUFFERS = 15
```

You can have up to 15 dBASE III PLUS files open; the remaining files are reserved for DOS and dBASE III PLUS system files. dBASE III PLUS uses 15 disk buffers. This value depends on the amount of memory in your computer. For example, 256K computers can only have 4 disk buffers allocated to dBASE III PLUS.

The information in Config.sys is accessed by your computer when you start it up. Therefore, Config.sys must reside in the disk or directory from which your computer boots.

## Creating and Modifying Config.sys

Config.sys is a standard ASCII text file that you create or modify with any word processor or line editor (such as EDLIN in DOS). You can also use MODIFY COMMAND in dBASE III PLUS.

Using MODIFY COMMAND, you would enter MODIFY COMMAND Config.sys at the dot prompt. The filename and menu of text editing commands appear at the top of the screen. Type the configuration commands and their values. Each command in the file must be on a separate line, as shown in the preceding section, and followed by a ↵. To save the file, press **Ctrl-W**.

Store Config.sys in the disk or directory from which your system boots. If you start your computer with a floppy disk, put Config.sys on this disk. If you boot from a hard disk, put Config.sys in the root directory.

The information in Config.sys is accessed by your computer when you start it up. So, after creating the file for the first time, and each time you change the file, you must *always* reboot your computer to use the new values.

> **TIP**
> You can make System Disk #1 a boot disk to start your computer. See *Getting Started* for instructions.

## Config.db

Config.db is a dBASE configuration file that is used to customize your workspace in dBASE III PLUS. The dBASE configuration commands are: BUCKET, COMMAND, GETS, MVARSIZ, PROMPT, TEDIT, and WP. The Config.db file can also contain new values for most of the SET commands and the function keys.

When you receive dBASE III PLUS, all of these commands have default or preset values. You reset them by entering the commands and their new values in Config.db. For example, the Config.db file on System Disk #1 contains values for two commands:

```
STATUS = ON
COMMAND = ASSIST
```

The default value for SET STATUS is normally OFF, but STATUS = ON resets it, which means you see the status bar while working with dBASE III PLUS. COMMAND = ASSIST means that instead of dBASE III PLUS starting up at the dot prompt, as it would normally, The Assistant appears.

You may modify Config.db and remove or add to these commands. The information you enter can only be accessed when you start dBASE III PLUS. Therefore, Config.db must reside on the disk and directory where you start dBASE III PLUS. Usually this is System Disk #1 or on a hard disk, in the same directory as dBASE III PLUS.

**NOTE**

1. You can store Config.db on a different drive or directory if you set up the DOS PATH command to find it.
2. You can also have different Config.db files for different data disks, directories, or applications. In this case, the DOS PATH command must be set up to find dBASE III PLUS, and a Config.db file must reside on the disk and directory where you wish to start dBASE III PLUS.

See your DOS manual for details on the PATH command.

The following sections discuss the dBASE configuration commands, the SET commands, and the function key values you may specify in Config.db. You will also learn how to create or modify Config.db.

## Configuration Commands

The dBASE configuration commands are only available in the Config.db file; you cannot enter them at the dot prompt. Table 4-1 lists these commands and an explanation of how to use them follows.

| Command Name | Value |
|---|---|
| BUCKET | The amount of memory allocated for the PICTURE and RANGE options of the @...GET command. The default value is 2, which stands for 2*1024 bytes. |
| COMMAND | Any dBASE III PLUS command to be executed when you start the program. |
| GETS | The number of @...GETs that may be active at one time. The default value is 128. |
| MVARSIZ | The maximum amount of space allocated to memory variables. The default value is 6,000 bytes. |
| PROMPT | The dBASE III PLUS prompt. The default is a dot (.). |
| TEDIT | The external word processor to be used with MODIFY COMMAND. |
| WP | The external word processor to be used with Memo fields. |

Table 4-1 dBASE configuration commands

### BUCKET

BUCKET can be a value from 1 to 31. Memory is allocated in kilobytes (1024 bytes). You would increase the default value of 2 kilobytes if you were creating or using a large format file with multiple screens and PICTURE clauses. You need to lower the value to 1 kilobyte for computers with only 256K of memory. (See the 256K Configuration section below.)

### COMMAND

The Config.db file on System Disk #1 contains the line COMMAND = ASSIST, which automatically displays The Assistant when you start the program. Remove this line to automatically display the dot prompt instead. Pressing **F2** at the dot prompt redisplays The Assistant, and pressing **Esc** at The Assistant main menu brings back the dot prompt.

The value that you specify for COMMAND can be any valid dBASE III PLUS command. If you set COMMAND = DO <filename>, the named file is immediately executed when you start the program.

> **NOTE**
> You can override the value for COMMAND by executing a command (.prg) file from the DOS prompt. You would enter dbase <program filename>. The command file is executed immediately after starting dBASE III PLUS.

### GETS

The GETS setting is similar to BUCKET. You would increase the value for GETS if you were creating or using a large format file with multiple screens. You would lower it for computers with only 256K memory. (See the 256K Configuration section below.)

### MVARSIZ

You would increase the default value of MVARSIZ if you had a large program with many variables. You would lower the value to 3 (for 3,000 bytes) to run dBASE III PLUS on computers with only 256K.

### PROMPT

The PROMPT setting allows you to replace the dot prompt with a string up to 19 characters long. You may use any character that you can display on your screen, including graphics characters. Avoid using characters which send commands to your printer.

### TEDIT and WP

To use a word processor other than the dBASE III PLUS text editor for either TEDIT or WP, you must have enough extra memory to load the Command.com file and the word processor(s). The word processor specified with TEDIT is used with MODIFY COMMAND; the word processor specified with WP is used to enter memo fields.

By specifying another word processor for WP, you can have memo fields larger than the default maximum of 5,000 bytes. The word processor(s) should reside in the same directory as dBASE III PLUS. Command.com can be in a separate directory, but you must specify a path to find it using DOS SET COMSPEC. For example, you would enter `SET COMSPEC = C:\Command.com` at the DOS prompt.

## SET Commands

You can include values for most of the SET commands in Config.db. The SET commands control the dBASE III PLUS environment. For example, you can SET BELL OFF so you don't hear the beep. If you have a color monitor, you can use SET COLOR to specify colors for your screen display. You can define specific drives or directories using SET PATH TO. If dBASE III PLUS cannot find a file in the current directory, it searches the designated drives or directories.

Table 4-2 shows which SET commands you can implement in Config.db. Chapter 5 describes them briefly in the Classes of Commands section under Parameter Controls. For more information on the default value, purpose, and use of a SET command, go to the reference section, also in Chapter 5.

## Function Keys

You can program the **F2** through **F9** keys to execute dBASE commands. The **F1** key is reserved for getting help in dBASE III PLUS. The function keys have these preset values:

| | | | |
|---|---|---|---|
| **F1** | — HELP | **F6** | — DISPLAY STATUS |
| **F2** | — ASSIST | **F7** | — DISPLAY MEMORY |
| **F3** | — LIST | **F8** | — DISPLAY |
| **F4** | — DIR | **F9** | — APPEND |
| **F5** | — DISPLAY STRUCTURE | **F10** | — EDIT |

You can change them to execute different commands or a command (.prg) file with DO < filename >, or use them to display a character string.

## Creating and Modifying Config.db

Config.db is an ASCII text file you create with a word processor or line editor. You can also use MODIFY COMMAND in dBASE III PLUS.

Using MODIFY COMMAND, you would enter MODIFY COMMAND Config.db at the dot prompt. The filename and menu of text editing commands appear at the top of the screen. Type the commands and the desired values, using this format:

< command name > = < value >

This format does not require the word SET when defining a SET option and does not require delimiters.

You use a slightly different format to set function key values:

function = < function key number >, < expC >

or F < n > = < expC >

The character string you enter must be delimited with single quotes, double quotes, or square brackets. To specify the ↵ key in the character string, use a semicolon; for example,
function = 2, 'BROWSE;'

Each command in the file must be on a separate line, and followed by a ↵. To save the file, press **Ctrl-W**.

Because Config.db is read only when you start dBASE III PLUS, you must store Config.db on the disk and directory where you start dBASE III PLUS. This is usually System Disk #1 or on a hard disk, in the same directory as dBASE III PLUS.

**NOTE**

1. You can store Config.db on a different drive or directory if you set up the DOS PATH command to find it.
2. You can have different Config.db files for different data disks, directories, or applications. In this case, the DOS PATH command must be set up to find dBASE III PLUS, and a Config.db file must reside on the disk and directory where you wish to start dBASE III PLUS.

   See your DOS manual for details on the PATH command.

For the same reason, you also need to restart dBASE III PLUS after creating the file for the first time and after each time you change the file, if you wish to use the new values.

When starting dBASE III PLUS, if Config.db has an incorrect command entry or value, dBASE III PLUS will display an error message, but continue reading the file. Once in the program, you can correct the file, following the instructions above.

You can override any of the Config.db entries except the configuration commands at the dot prompt. Simply type the command with its new value. Remember that the syntax for SET commands includes the word SET at the dot prompt. Reset the function keys with the SET FUNCTION command. The new values remain until you change them or QUIT dBASE III PLUS. If you QUIT, the next time you run the program, the Config.db values will be in effect again.

| Command Name | Value | Default |
|---|---|---|
| *Configuration* | | |
| BUCKET | 1 to 31 | 2 |
| COMMAND | <command> [command arguments] | --- |
| GETS | 35 to 1023 | 128 |
| MVARSIZ | 1 to 31K | 6 |
| PROMPT | <prompt string> | dot (.) |
| TEDIT | program used for MODIFY COMMAND | --- |
| WP | word processor for memo fields | --- |
| *Function Keys* | | |
| FUNCTION | <function key number>, <expC> or | |
| F <n> | <expC> | |
| F1 | | HELP |
| F2 | | ASSIST |
| F3 | | LIST |
| F4 | | DIR |
| F5 | | DISPLAY |
| F6 | | STRUCTURE |
| F7 | | DISPLAY STATUS |
| F8 | | DISPLAY MEMORY |
| F9 | | DISPLAY |
| F10 | | APPEND |
| | | EDIT |
| *SET Commands* | | |
| ALTERNATE | <filename> | --- |
| BELL | ON/OFF | ON |
| CARRY | ON/OFF | OFF |
| CATALOG | <filename> | --- |
| CENTURY | ON/OFF | OFF |
| COLOR | <color option list> | --- |
| CONFIRM | ON/OFF | OFF |
| CONSOLE | ON/OFF | ON |
| DEBUG | ON/OFF | OFF |
| DECIMALS | 0 to 14 | 2 |
| DEFAULT | <drive specification> | the logged drive |
| DELETED | ON/OFF | OFF |

Table 4-2 Config.db entries (continued)

| Command Name | Value | Default |
|---|---|---|
| DELIMITERS | ON/OFF | OFF |
| DELIMITERS | < one or two characters > | : (colon) |
| DEVICE | SCREEN/PRINT | SCREEN |
| ECHO | ON/OFF | OFF |
| ESCAPE | ON/OFF | ON |
| EXACT | ON/OFF | OFF |
| HEADING | ON/OFF | ON |
| HELP | ON/OFF | ON |
| HISTORY | 0 to 16,000 | 20 |
| INTENSITY | ON/OFF | ON |
| MARGIN | 1 to 254 | 0 |
| MEMOWIDTH | 8 to available memory | 50 |
| MENU | ON/OFF | ON |
| PATH | < path name > | --- |
| PRINT | ON/OFF | OFF |
| SAFETY | ON/OFF | ON |
| SCOREBOARD | ON/OFF | ON |
| STATUS | ON/OFF | OFF |
| STEP | ON/OFF | OFF |
| TALK | ON/OFF | ON |
| TYPEAHEAD | 0 to 32,000 | 20 |
| UNIQUE | ON/OFF | OFF |
| VIEW | < filename > | --- |

Table 4-2 Config.db entries

## 256K Configuration

dBASE III PLUS has different configuration requirements for computers with only 256K of memory. The necessary files, Confi256.sys and Confi256.db, are on System Disk #1. These files can have the same commands as described in the preceding sections, however, the preset values for several commands are lower.

Confi256.sys contains these commands:

```
FILES = 20
BUFFERS = 4
```

To accommodate a lesser amount of memory, notice that only four buffers are allocated for reading and writing information to and from the disk.

Confi256.db contains these commands:

```
COMMAND = ASSIST
STATUS = ON
BUCKET = 1
GETS = 35
MVARSIZ = 3
HISTORY = 10
TYPEAHEAD = 10
```

Notice that the values for BUCKET, GETS, and MVARSIZ are the maximum values for these commands and that they are lower than the default values. Ten is the optimum value for the history and type-ahead buffers on 256K computers.

Before you can use dBASE III PLUS, Confi256.sys and Confi256.db must be renamed to Config.sys and Config.db. Copy the original Config.sys and Config.db files to a formatted disk and keep it for future use. If you upgrade your computer's memory to 384K or higher, you can use these configuration files to customize the program.

For complete instructions on preparing Confi256.sys and Confi256.db to use with dBASE III PLUS, see Appendix A in *Getting Started*.

**NOTE**
The RUN/! command, which executes other software from within dBASE III PLUS and requires additional memory to store them, is not supported on 256K computers.

# Chapter 5
# COMMANDS

## Symbols and Conventions

Symbols are used in a specific way throughout Chapter 5 as part of the syntax of commands. Most of these symbols reflect common standards used with other programming languages. The symbols used and their meaning are listed below:

| Symbol | Use |
|---|---|
| < > | Angle brackets indicate that you must enter the enclosed item. When you enter the requested item, do not type the angle brackets. |
| [ ] | Square brackets indicate that the enclosed item is an optional part of the syntax. If you enter the enclosed item, do not type the brackets. |
| / | The slash indicates that there is more than one way to use the command. The different options are shown separated by a /. When you use a command, you enter only one of the options. |
| **?** | When you type a command followed by **?**, a pull-down menu appears containing a list of appropriate files from which you can choose.This is called the catalog query, and can only be used when a catalog is open. |
| **< expC >**<br>**< expN >**<br>**< expD >**<br>**< condition >** | These notations are used to represent expressions of different types (character, numeric, or date), or a logical condition. When you enter the expression, do not type the angle brackets. |
| [ . . .] | The ellipsis . . . indicates you may specify one or more elements of the type listed. For example, when multiple expressions can be entered, they may be shown as [ < exp > , . . .]. |
| **? , *** | These symbols are used to indicate wildcard characters which have the same meaning as they have in DOS. A ? character in a filename or extension indicates any valid character will match. A * character indicates any acceptable number of valid characters will match. |
| **&&** | Used to indicate a program code comment. |

## Limitations

The following limitations apply to using the commands in this chapter:

| Item | Limitations |
|---|---|
| **filenames** | Filenames may be up to eight characters long. They must begin with a letter and cannot contain embedded blank spaces. Letters, numbers, and underscore characters are permitted. |
| **file extensions** | You can specify a file extension of up to three characters. Letters and numbers are permitted. You can specify file extensions of your own choosing, or dBASE III PLUS will provide default file extensions according to the file type used. |
| **field names** | Field names may be up to ten characters long. They must begin with a letter and cannot contain embedded blank spaces. Letters, numbers, and underscore characters are permitted. |
| **expression** | A valid expression may contain fields, variables, functions, constants, and operators. Expressions may be used in commands and functions, or used to specify an index key expression. An index key expression can be up to 220 characters long and must evaluate to a single data type: character, numeric, or date. |
| **index key** | When an index key expression is evaluated, it produces a result called the index key. The maximum length of an index key is 100 characters. |

**TIP**
This chapter provides working examples of dBASE III PLUS commands using files from the Sample Programs and Utilities disk. All example results assume that you're working with an untouched copy of the original file.

# Classes of Commands

dBASE III PLUS commands can be categorized by the functions they perform. The commands in each category are listed below.

## User Assistance

The following commands give on-line information:

| | |
|---|---|
| ASSIST | Menu-driven Assistant; aids execution of dBASE III PLUS commands. |
| DIR | Shows files present on the designated disk drive or directory. |
| DISPLAY MEMORY | Displays information about current memory variables. |
| DISPLAY STATUS | Displays information about the files currently in USE and system parameters. |
| DISPLAY STRUCTURE | Displays structure of a database file in USE. |
| HELP | Menu-driven; explains dBASE III PLUS commands and provides other information. |

## Creation of Files

The following commands create database and other associated files:

| | |
|---|---|
| COPY | Copies database file in USE to a new file. |
| COPY FILE | Duplicates any closed file. |
| CREATE | Creates a new database file. |
| IMPORT | Creates database and format files from pfs:File forms and creates a view file which includes both the database file and the format file. |
| INDEX | Creates an index file. |
| JOIN | Combines specified records and fields from two database files. |
| CREATE LABEL | Creates a label form file. |
| CREATE QUERY | Creates a query file. |
| CREATE REPORT | Creates a report form file. |
| CREATE SCREEN | Creates a screen file, a format file, and optionally a database file. |
| CREATE VIEW | Creates a view file. |
| EXPORT | Creates pfs:File forms from a dBASE III PLUS database file, using a dBASE III PLUS format file if one is open. |
| MODIFY COMMAND | dBASE III PLUS word processor; creates or modifies a command, format, procedure, or text file. |
| MODIFY STRUCTURE | Modifies a database file structure; creates a backup of the original database file. |
| SAVE | Copies current memory variables to a memory file. |
| SET CATALOG TO | Creates a new catalog or selects an existing catalog. |
| SORT | Creates a sorted version of the current database file in use. |
| TOTAL | Sums numeric fields of the current database file, creating a new database file with the resulting value totals. |

## Addition of Data

The following commands allow the addition of new data records to databases:

| | |
|---|---|
| APPEND | Allows addition of data at the end of a database file. |
| BROWSE | Allows addition of data at the end of a database file. |
| INSERT | Allows insertion of data in a database file. |

## Editing of Data

The following commands allow editing of the data within a database:

| | |
|---|---|
| BROWSE | Allows editing of data within a full-screen window display. |
| CHANGE | Allows editing of specified fields and records in database files. |
| EDIT | Allows editing of specified fields and records in database files (same as CHANGE). |
| READ | Permits entry to fields or variables in a GET command. |
| REPLACE | Replaces data fields with specified values. |
| UPDATE | Allows batch updates of a database file. |

## Deletion of Data

The following commands control deletion of data.

| | |
|---|---|
| DELETE | Marks records for deletion. |
| PACK | Removes records marked for deletion. |
| RECALL | Unmarks records marked for deletion. |
| ZAP | Removes all records from the active database file. |

## Data Display

The following commands display selected data from database files.

| | |
|---|---|
| @...SAY...GET | Displays user-formatted data on the screen or printer. |
| ? | Displays an expression list. |
| AVERAGE* | Computes and displays the arithmetic mean of one or more field expressions for the database records specified. |
| BROWSE | Displays up to 17 records per screen with as many fields as will fit on the screen. |
| COUNT* | Counts and displays the number of records matching a specified criteria. |
| DISPLAY | Displays records and fields in table form. |
| LABEL | Displays record data in label form. |
| LIST | Lists records and fields in table form. |
| REPORT | Displays data in report form. |
| SUM* | Computes and displays the sum of selected numeric fields. |

* These commands do not display results if SET TALK is OFF.

## Positioning the Record Pointer

The following commands position the record pointer within database files.

| | |
|---|---|
| CONTINUE | Positions the record pointer to the next record meeting conditions specified by the LOCATE command. |
| FIND | Positions the record pointer to the record matching the specified character string or numeric value. |
| GO/GOTO | Positions the record pointer to a specific record, regardless of the record pointer's current position. |
| LOCATE | Positions the record pointer to a record matching a specified condition. |
| SEEK | Positions the record pointer to a record matching a specified expression. |
| SKIP | Positions the record pointer forward or backward a specified number of records relative to the current record pointer position. |

## Manipulating Database Files

The following commands affect entire database files:

| | |
|---|---|
| APPEND FROM | Adds records from other files to the dBASE III PLUS file currently selected. |
| CLOSE | Closes all files of a specified type. |
| COPY | Copies the currently selected database file to another file. |
| ERASE/DELETE FILE | Deletes files from the directory. |
| MODIFY STRUCTURE | Alters a database file structure. |
| REINDEX | Rebuilds existing index files. |
| RENAME | Renames a file. |
| SELECT | Switches among up to 10 open database files or work areas. |
| SORT | Creates a copy of a database file sorted on one or more data fields. |
| USE | Specifies the database file to be used for all operations until another USE command is issued. |

## Manipulating Other Types of Files

The following commands modify dBASE program files or other database-related dBASE III PLUS files.

| | |
|---|---|
| MODIFY COMMAND | Word processor used to edit dBASE program, screen format, and other ASCII text files. |
| MODIFY LABEL | Edits a label form file. |
| MODIFY QUERY | Modifies a query file. |
| MODIFY REPORT | Modifies a report form file. |
| MODIFY SCREEN | Modifies a screen file, a format file, and optionally a database structure. |
| MODIFY VIEW | Modifies a view file. |

## Using Memory Variables

The following commands manipulate memory variables:

| | |
|---|---|
| ACCEPT | Stores a character string in a memory variable. |
| AVERAGE | Stores the arithmetic mean of numeric expressions to memory variables when a memory variable list is specified. |
| CLEAR ALL | Erases all current memory variables, closes all database files, and selects work area 1. |
| CLEAR MEMORY | Clears all current memory variables. |
| COUNT | Stores results of COUNT into memory variables. |
| DISPLAY MEMORY | Displays current memory variables. |
| INPUT | Stores the result of an expression into a memory variable. |
| PARAMETERS | Specifies optional parameters to be used by a program. |
| PRIVATE | Hides memory variables from higher-level dBASE program files. |
| PUBLIC | Defines a memory variable as a global variable. |
| READ | Allows data entry into an @...GET variable. |
| RELEASE | Clears specified memory variables. |
| RESTORE | Retrieves memory variables stored in a disk file. |
| SAVE | Saves the current memory variables to a disk file. |
| STORE | Stores the result of an expression in a memory variable. |
| SUM | Stores sums of numeric expressions into memory variables when a memory variable list is specified. |
| WAIT | Accepts a single input character and stores the character in a memory variable. |

## Programming

The following commands assist in the control and usage of dBASE program files:

| | |
|---|---|
| ACCEPT | Allows input of character data into a memory variable. |
| CANCEL | Terminates dBASE program file execution. |
| DO | Executes a program or procedure file. |
| DO WHILE | Executes a structured loop in programs. |
| EXIT | Leaves a DO WHILE loop. |
| LOOP | Skips to the beginning of a DO WHILE loop. |
| ENDDO | Terminates a DO WHILE loop. |
| DO CASE | Sets up a structure for executing one of several specified blocks of program commands. |
| CASE | Marks the beginning of each block of program commands in a DO CASE command. |
| OTHERWISE | Alternative to all other CASEs in DO CASE. |
| ENDCASE | Terminates a DO CASE command. |
| IF | Allows conditional execution of commands. |
| ELSE | Allows an alternate path of execution within IF. |
| ENDIF | Terminates an IF command. |
| INPUT | Allows input of an expression result in a memory variable. |
| MODIFY COMMAND | Word processor used to edit dBASE program files. |
| PROCEDURE | Defines the beginning of a procedure in a procedure file. |
| QUIT | Closes all files and exits dBASE III PLUS. |
| RETRY | Returns to calling program and re-executes the command that caused the called program to execute. |
| RETURN | Returns control to calling program and executes the next command. |
| TEXT | Displays a block of text data in a command file. |
| ENDTEXT | Specifies the end of a block of TEXT data in a command file. |
| WAIT | Suspends program processing until a key is entered. |

## Environmental Commands

The following commands control the dBASE III PLUS environment and peripheral devices:

| | |
|---|---|
| CLEAR | Clears the display screen. |
| EJECT | Ejects a page on the printer. |
| SET | Menu used to set dBASE session control parameters. |

## Parameter Controls

The following commands set dBASE III PLUS system control parameters (upper case indicates default setting):

| | |
|---|---|
| SET ALTERNATE on/OFF | Controls recording of dBASE commands and output in a text file. |
| SET ALTERNATE TO | Names a text file for saving dBASE commands and output (or closes the file). |
| SET BELL ON/off | Determines whether the bell rings during data entry (at the end of a field or when an error occurs). |
| SET CARRY on/OFF | Determines whether the last record's contents are carried over into an APPENDed record. |
| SET CATALOG on/OFF | Determines whether new files are added to an open catalog. |
| SET CATALOG TO | Creates, opens, or closes a catalog. |
| SET CENTURY on/OFF | Determines whether a date displays the century. |
| SET COLOR ON/OFF | Sets output to a color or monochrome monitor. The default is the one in use when you start dBASE III PLUS. |
| SET COLOR TO | Sets screen display attributes. |
| SET CONFIRM on/OFF | Determines whether a carriage return is required at the end of a field entry. |
| SET CONSOLE ON/off | Determines whether output from commands appear on the screen. |
| SET DATE | Sets the format for date expressions. |
| SET DEBUG on/OFF | Determines whether output of dBASE commands, with SET ECHO ON, is sent to a printer. |
| SET DECIMALS TO | Determines the minimum number of decimals that will be displayed in calculation results. |
| SET DEFAULT TO | Specifies the default drive for a file search. |
| SET DELETED on/OFF | Determines whether certain commands ignore or process records marked for deletion. |

| | |
|---|---|
| SET DELIMITERS on/OFF | Determines whether specified delimiters are used to enclose an entry field. |
| SET DELIMITERS TO | Specifies the delimiters for entry fields when SET DELIMITER is ON. |
| SET DEVICE TO SCREEN/print | Sends the results of @...SAY commands to the screen or printer. |
| SET DOHISTORY on/OFF | Determines whether commands executed in a program file are recorded in a history buffer. |
| SET ECHO on/OFF | Determines whether program file commands are echoed to the screen. |
| SET ESCAPE ON/off | Determines whether pressing **Esc** suspends command execution. |
| SET EXACT on/OFF | Determines whether an exact match is required when comparing character strings. |
| SET FIELDS on/OFF | Determines whether a specified field list is used. |
| SET FIELDS TO | Specifies a field list for subsequent commands. |
| SET FILTER TO | Filters out all database records that do not satisfy a condition in a query file or expression. |
| SET FIXED on/OFF | Determines whether a fixed number of decimal places are displayed. |
| SET FORMAT TO | Opens a format file for data entry. |
| SET FUNCTION TO | Sets function key values. |
| SET HEADING ON/off | Specifies whether field names appear above the fields with the LIST or DISPLAY commands. |
| SET HELP ON/off | Determines whether a prompt offering help appears when an error occurs. |
| SET HISTORY ON/off | Determines whether commands are recorded in a history buffer. |
| SET HISTORY TO | Specifies the maximum number of executed commands that can be stored in the history buffer. |
| SET INDEX TO | Opens named index files. |
| SET INTENSITY ON/off | Determines whether full-screen operations such as APPEND or EDIT use the enhanced screen attribute (inverse video) to display fields. |
| SET MARGIN TO | Sets the left margin of output to a printer. |
| SET MEMOWIDTH TO | Sets the column width of memo field output. |
| SET MENUS ON/off | Determines whether a menu appears at the top of the screen during full-screen commands. |

| | |
|---|---|
| SET MESSAGE TO | Specifies a user-defined message to display on the bottom of the screen (when SET STATUS is ON). |
| SET ODOMETER TO | Specifies when the record count display is updated. |
| SET ORDER TO | Sets up any open index file as the master or controlling file without closing and reopening the index files. |
| SET PATH TO | Specifies a path for a file search. |
| SET PRINT on/OFF | Determines whether output is sent to a printer. |
| SET PRINTER TO | Redirects output to the specified device. |
| SET PROCEDURE TO | Opens a named procedure file. |
| SET RELATION TO | Links two databases according to a key expression. |
| SET SAFETY ON/off | Determines whether a prompt is displayed when a file is about to be overwritten. |
| SET SCOREBOARD ON/off | Determines whether dBASE III PLUS messages appear on line 0 if the status bar is off. |
| SET STATUS on/OFF | Determines if the status bar is displayed below the dot prompt. |
| SET STEP on/OFF | Specifies whether execution stops after each command in a dBASE program file is executed. |
| SET TALK ON/off | Determines whether dBASE command responses are displayed on the screen. |
| SET TITLE ON/off | Determines whether a file title prompt appears when a new file is added to a catalog. |
| SET TYPEAHEAD TO | Specifies the size of the type-ahead buffer. |
| SET UNIQUE on/OFF | Determines whether all records with identical index keys, or just the first record, appear in an index file. |
| SET VIEW TO | Opens a view (.vue) file relating one or more database files. |

## External Program Interfacing

The following commands allow you to interface programs written in other languages with dBASE III PLUS:

| | |
|---|---|
| CALL | Executes a binary file (module) placed in memory with the LOAD command. |
| LOAD | Places a binary file in memory where it can be executed with the CALL command. |
| RUN | Executes a specified DOS command, or any other program executable from DOS, from within dBASE III PLUS. |

## Debugging Commands

The following commands allow you to keep track of commands being executed by dBASE III PLUS:

| | |
|---|---|
| DISPLAY HISTORY | Displays the commands stored in the history buffer. |
| LIST HISTORY | Lists the commands stored in the history buffer. |
| RESUME | Causes a SUSPENDed program to resume execution. |
| SET DEBUG on/OFF | Determines whether output of dBASE commands, with SET ECHO ON, is sent to a printer. |
| SET DOHISTORY on/OFF | Determines whether commands executed in a program file are recorded in the history buffer. |
| SET ECHO on/OFF | Determines whether program file commands are echoed on the screen. |
| SET HISTORY TO | Specifies the number of executed commands to store in the history buffer. |
| SET STEP on/OFF | Specifies whether execution stops after each command in a dBASE program file is processed. |
| SUSPEND | Halts execution of the currently executing command file or procedure and displays the dot prompt. |

## Event Processing

The following commands causes a specifed action to occur when a specific condition takes place, regardless of what other task dBASE III PLUS is doing at the time:

| | |
|---|---|
| ON ERROR | Executes the specified command when an error condition occurs |
| ON KEY | Executes the specified command when any key is pressed. |
| ON ESCAPE | Executes the specified command when **Esc** is pressed. |

## ?/??

?/?? evaluates and displays the value of dBASE III PLUS expression lists. ?/?? roughly translates to: "What is ...?" or "What is the value of ...?"

## Syntax:

?/?? < expression list >

## Usage

The single question mark issues a carriage return and line feed before the expression list is displayed. The double question mark does not. When used in a program, ?? displays the expression list beginning with the current cursor or printer position. In the interactive mode (at the dot prompt), this overwrites the command on the screen.

## Tips

? without a following expression displays a blank line and can be used to skip a line in the output. ?/?? may be used with SET PRINT ON and the CHR() function to send escape code sequences to your printer, for example, to set bold or italic printing modes.

## Examples

The Clients database file on the Sample Programs and Utilities disk is used to illustrate the command. To display the contents of the Lastname and City fields from the current record:

```
. USE Clients
. ? Lastname,City
Buckman                  Oxnard
```

To determine if the current record belongs to Butler:

```
. ? Lastname = 'Butler'
.F.
```

dBASE III PLUS returns .F. for false, .T. for true.

To display the results of a calculation:

```
. ? 2 * 4 + (9/3)
  11.00
```

To display the system date:

```
. ? DATE()
05/15/84
```

## @

@ is used to create custom forms for data input and output. It displays information in a specified format at a given set of coordinates.

## Syntax

@ <row, col> [[SAY <exp> [PICTURE <clause>]]
[GET <variable> [PICTURE <clause>]
[RANGE <low>, <high>]]]/[CLEAR]

## Usage

Row and col are numeric expressions. For a 24x80 terminal, the row coordinate can range from 0 to 23 and the column coordinate from 0 to 79.

With SET STATUS ON, line 22 is reserved for the status line. If you SET STATUS OFF and do not SET SCOREBOARD OFF, line 0 is used for status information display. To free these lines for other use, you must SET STATUS OFF and SET SCOREBOARD OFF.

You can use SET DEVICE TO PRINT to route @...SAY commands to the printer. @...GETs are not routed to the printer. Most printers have limitations on the row and column coordinates. When sending @ commands to a printer, decreasing the row number in consecutive @ commands causes a page eject. Similarly, if two @ commands have the same row coordinate, the second one must have a larger column coordinate.

The @...SAY command displays information that you do not want to change. The value of any valid dBASE III PLUS expression except memo fields can be displayed.

The @...GET command displays and allows editing of information from existing memory variables and fields. The READ command activates a full-screen editing mode which allows editing of GET fields.

Besides using @...SAY...GETs with the READ command, you can create a format (.fmt) file containing @...SAY...GETs. You can use a format file with the APPEND, EDIT, and READ commands. If you create a format file, you can also specify memo fields with the GET command.

@ row,col CLEAR erases the screen below and to the right of the coordinates.

@ row,col erases a line beginning at the column position specified.

## Options

The RANGE option is used with numeric and date variables to specify lower and upper bounds. <low> and <high> are expressions which define the minimum and maximum values that may be entered in response to the GET.

Since there is no literal representation for dates, you must use the CTOD( ) function to use constant dates in a RANGE. For example, RANGE CTOD('01/01/85'), CTOD('12/31/85') excludes all dates with a year other than 1985. Enter one expression if you want to specify only a lower or upper limit. You must include the comma (,) as in RANGE ,30 or RANGE 10, . If an invalid number or date is then entered, dBASE III PLUS prompts for a new value until a valid number or date is entered.

If you specify a RANGE and press ↵ in response to a GET, no range checking is done. Therefore, if a field or variable is set to a value outside the specified range, pressing ↵ leaves the value unchanged.

The PICTURE option allows formatted output capability and restricts the type of data that may be entered into a variable. The clause may consist of a *function* and/or a *template*, and must be delimited.

### Functions

If a PICTURE function is used, the @ symbol must appear as the first character in the clause. If a function is used in conjunction with a template, a space must separate the two.

dBASE III PLUS provides the following functions:

| Symbol | Description |
|---|---|
| **C** | Displays CR (credit) after a positive number. |
| **X** | Displays DB (debit) after a negative number. |
| **(** | Encloses negative numbers in parentheses. |
| **B** | Left-justifies numeric data. |
| **Z** | Displays zero numeric value as a blank string. |
| **D** | American date format. |
| **E** | European date format. |
| **A** | Alphabetic characters only. |
| **!** | Allows any character and converts letters to upper case. |
| **R** | Literal characters displayed in the template, but not entered in the field. |
| **S <n>** | Limits field width display to <n> characters and horizontally scrolls the characters within it in <n> columns. <n> must be a literal positive integer. |

Some functions are restricted according to the data types to which they apply. The functions C, X, (, B, and Z can be used with numeric data only. Furthermore, C, X, and ( can only be used with the SAY clause. D and E apply to date, character, and numeric data. A, !, R, and S are only relevant to character data.

New functions may be defined by logically combining the functions in the table. For example, the function XC displays DB after negative numbers and CR after positive numbers. However, some functions cannot be used together, such as D and E or X and (.

S <n> is used to display and edit a character GET variable in a number of columns that is less than the actual length of the character string. The number of columns is defined by a literal integer <n>, and no spaces are allowed between the 'S' and the integer.

The string scrolls within the specified width, allowing you to view and edit the entire character string. Use →, ←, **Ctrl**-F, and **Ctrl**-A to bring hidden characters into view. When entering data, the string scrolls automatically.

**Templates**

A template is formed by using a single symbol for each character to be displayed or input. Any character may be used, but characters other than the template symbols have no special meaning.

If the R picture function is used with a template containing characters other than template symbols, those characters are inserted into the display and not stored as part of the GET variable. If R is not used, those characters are displayed rather than the corresponding string character and are stored in the GET variable. R applies only to character type variables. For numeric variables, non-template symbols are always inserted into the display and never stored as part of the number. Avoid using non-template symbols for date and logical variables.

dBASE III PLUS provides the following template symbols:

| Symbol | Description |
|---|---|
| **9** | Allows only digits for character data. Allows digits and signs for numeric data. |
| **#** | Allows only digits, blanks, and signs. |
| **A** | Allows only letters. |
| **L** | Allows only logical data. |
| **Y** | Allows only logical Y, y, N, n. Converts y and n to upper case letters. |
| **N** | Allows letters and digits. |
| **X** | Allows any character. |
| **!** | Converts letters to upper case and has no affect on other characters. |
| **$** | Displays dollar signs in place of leading zeroes. |
| ***** | Displays asterisks in place of leading zeroes. |
| **.** | Specifies decimal position. |
| **,** | Displays if there are digits to left of the comma. |

Symbols !, 9, #, A, N, and X may be used effectively with SAYs and GETs. Symbols 9, #, A, N, and X do not prevent undefined characters from being displayed.

If a PICTURE template is used to GET a decimal number, the decimal point must be included in the template. The template must also leave room for at least one digit to the left of the decimal point, and leave room for the sign, if you want to use the minus sign.

## Tips

To design a custom form, use CREATE SCREEN or MODIFY COMMAND to create a format file (.fmt) that consists of @ commands.

Activate the format file with SET FORMAT TO <.fmt filename>. You can append new records or revise existing ones from a .fmt file using any of the full-screen editing commands such as EDIT and APPEND.

## Programming Notes

If you want to use a multi-page format (.fmt) file in which the @...SAY...GETs continue on from 2 to 32 screens, include a READ wherever you want a page break. The **PgDn** and **PgUp** keys flip the pages. Multi-page format files only work when the .fmt file is opened with SET FORMAT TO.

## Examples

To display the information in the first record of the Clients file:

```
. USE Clients
. CLEAR
. @ 5,0 SAY TRIM(Firstname)+ ' ' +Lastname
. @ 7, 0 SAY Address
. @ 9, 0 SAY TRIM(City)+ State + Zipcode
```

As you enter each command, the resulting expression is displayed on your screen. For this example, the results displayed on your screen should be the following:

```
Claire Buckman

8307 Santa Anita Blvd

OxnardCA93034
```

For a more readable display, specify the spaces in quotes as part of the expression, as in the first @...SAY expression in the above example. Notice in the example above that the last display does not include spaces, and is therefore not as readable as the first display in the example.

To display the address using all capitals:

```
. @ 11,0 SAY Address PICTURE '@!'
8307 SANTA ANITA BLVD
```

To illustrate how to format numeric output:

```
. number1 = 1238754
1238754
. number2 = 54.98
54.98
. number3 = -2.56
-2.56
. clear
```

@...GET...SAY commands will display the formatted output at the specified coordinates. Type the following commands:

```
. @ 1,0 SAY number1
. @ 2,0 SAY number1 PICTURE '$$$,$$$,$$$,$$$'
. @ 3,0 SAY number2 PICTURE '***,***,***.***'
. @ 4,0 SAY number3 PICTURE '@X ***,***.**'
```

The lines displayed on your screen from these commands are the following:

```
1238754
$$$$$$1,238,754
*********54.980
******2.56 DB
```

The $ and * template symbols use commas in the same way. Both display commas only when they are needed to separate digits within a number.

```
. @ 5,15 CLEAR
```

This command erases the screen below the 5th row and 15th column, and releases all GETS in the area erased.

## See Also

APPEND, CHANGE, COL(), CREATE/MODIFY SCREEN, EDIT, INSERT, MODIFY COMMAND, PCOL(), PROW(), READ, ROW(), SET BELL, SET CONFIRM, SET DELIMITERS, SET DEVICE, SET FIELDS, SET FORMAT, SET INTENSITY

## @...TO

@...TO draws or clears a box with single or double lines.

### Syntax

@ <row1,col1> [CLEAR ] TO <row2,col2> [DOUBLE]

### Defaults

Unless otherwise specified, a single line delineates the box.

### Usage

The @...TO command draws a box.

<row1,col1> are the coordinates of the upper left corner of the box.

<row2,col2> are the coordinates of the lower right corner of the box.

If the row coordinates are the same, a horizontal line is drawn. If the column coordinates are the same, a vertical line is drawn.

### Options

DOUBLE draws a double-line box rather than the default single- line one.

CLEAR erases the screen in the area starting at <row1,col1> up to and including <row2,col2>. It clears both the box and its contents.

### Programming Notes

You can use the @...TO command from the dot prompt or in a command or format file.

Use it with the @S<n> function of the @...SAY...GET command to limit the size of the input area on the screen. The @S<n> function allows horizontal scrolling within the input area.

## Example

You can draw two boxes next to each other, put information in each, and then clear part of the screen. Enter the following information into a command (.prg) file using MODIFY COMMAND.

```
. MODIFY COMMAND Line
M1 = 'Test scrolling'                  && Create a memvar
@ 1,1 TO 10,15 DOUBLE                  && Draw a double line box
@ 3,35 TO 7,69                         && Draw a single line box
@ 3,3 GET M1 PICTURE '@S8'             && Display 8 characters
@ 4,38 SAY 'Press Control-F '
@ 5,38 SAY 'to scroll; Press '
@ 6,38 SAY 'ENTER to finish.'
READ SAVE                              && Edit @S and save display
@ 3,35 CLEAR TO 7,69                   && Erase single line box
READ                                   && Edit remaining box
CLEAR                                  && Erase the entire screen
```

Press **Ctrl-End** to save the .prg file. To execute the program, type DO Line↵ at the dot prompt.

## See Also

@...SAY...GET

## ACCEPT

ACCEPT is used primarily in command files to prompt a user for keyboard entry. It creates a character memory variable in which it stores the keyboard entry. Terminate data entry with ↵.

## Syntax

ACCEPT [ < prompt > ] TO < memvar >

## Usage

The prompt may be a character type memory variable or a character string. If a character string, it must be delimited by 'single quotes', "double quotes", or [brackets].

The keyboard entry does not require delimiters: ACCEPT treats all user input as character-type data.

If ↵ is entered in response to the ACCEPT command, the content of the memory variable is null (without any contents, or ASCII 0).

A maximum of 254 characters can be entered into a variable with ACCEPT.

## Example

To prompt the user to "Enter your social security number:" and store the keyboard entry in the Ssno memory variable:

```
. ACCEPT "Enter your social security number:" TO Ssno
Enter your social security number:
```

## See Also

INPUT, WAIT

## APPEND

APPEND allows you to add new records to the end of the active database file. It is the primary command for full-screen data entry.

### Syntax

APPEND [BLANK]

### Usage

APPEND places you in the full-screen data entry mode. One new record at a time is presented as a screen "form" for data entry.

Terminate the process by pressing ↵ when a new record is presented, or by pressing **Ctrl-End**. If you move to a new record and press **Ctrl-End**, a blank record is added to the file.

The **PgUp** key moves the cursor to previous records, enabling you to edit them. They will be in record order, not indexed order. The **PgDn** key moves forward through the records, and returns to the APPEND mode after the last record is indexed.

To enter data into a memo field, press **Ctrl-PgDn** when the cursor is positioned at the beginning of the field. A blank screen appears and text can be entered. The control keys are the same as for MODIFY COMMAND. A cursor control menu can also be toggled on and off with **F1**. To exit a memo field, press **Ctrl-End** (or **Ctrl-W**) to save your entry in the memo field or press **Esc** to abandon the entry.

APPEND allows you to add records to a single database file only. Using a format file, it is possible to do @...GETs for fields from several related files. Using APPEND with a format file like this does not add records to unselected files.

### Options

The BLANK option adds a blank record to the end of the database file, but the full-screen mode is not entered. The BLANK record becomes the current record.

### Tips

If index files are active during APPEND, any new record will be added to the index file in the appropriate location.

The directory entry for the active database file may not reflect all new records until the file is closed.

## Example

The Clients database file on the Sample Programs and Utilities disk is used for the following example. To enter the full-screen data entry mode and begin APPENDing records to the database, type the following at the dot prompt:

```
. SET STATUS ON
. USE Clients
. APPEND
```

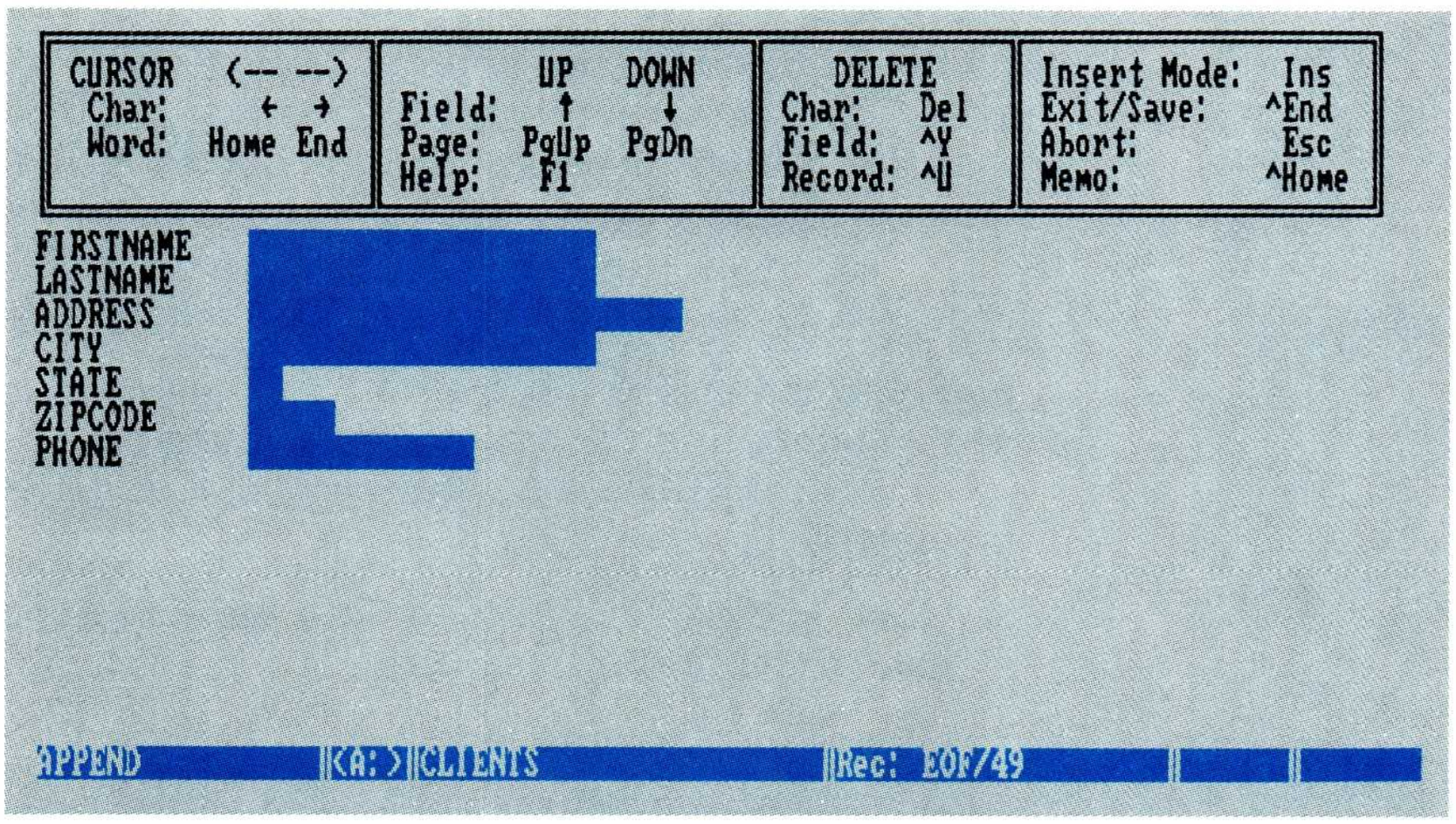



Figure 5-1 The APPEND command

Press **Ctrl-End** to exit and save all changes, or **Esc** to exit and save changes to all but the current record.

## See Also

MODIFY COMMAND, SET CARRY, SET FORMAT TO

## APPEND FROM

APPEND FROM copies records from an existing file to the end of the active database file. The FROM file does not have to be a dBASE III PLUS file.

### Syntax

APPEND FROM <filename/?> [FOR <condition>]
[[TYPE] <file type>]

### Defaults

The filename must include the drive designator if the file is not on the default drive unless a path is set to that drive.

If a file extension is not entered as part of the filename, and the file type is not specified, dBASE III PLUS assumes the .dbf extension.

If a file extension is not entered as part of the filename and the SDF or DELIMITED option is specified, dBASE III PLUS assumes a.txt extension. Other file types are assumed to have the default extensions supplied by their respective software packages.

### Usage

If the FROM file is a dBASE III PLUS database (.dbf) file:

- Records marked for deletion *are* appended and are not marked for deletion in the active database if SET DELETED is OFF. If SET DELETED is ON, only records not marked for deletion are appended.
- Only field names found in both files are appended. They do not have to be in the same order.
- Do *not* specify a file type.

If a field in the FROM file is larger than the same field in the active database, excess character data is lost, and excess numeric data is replaced by asterisks.

The FOR expression refers only to field names that are common to both files.

## Options

The options for < file type > are:

- DELIMITED [WITH BLANK/ < delimiter > ] — Delimited Format ASCII file. Data is appended character by character starting on the left. Each record must end with a carriage return and line feed. A comma separates each field and, in addition, double quotation marks surround character data unless you specify another delimiter. This is the same as DELIMITED WITH " .

  DELIMITED WITH BLANK works with files containing fields separated by one space.

  DELIMITED WITH < delimiter > works with files containing fields separated by commas and character strings also enclosed in specified delimiters. The default delimiter is the double quote character.

- SDF — System Data Format ASCII file. Data is appended character by character starting on the left. Each record is the same length and ends with a carriage return and line feed.

- DIF — VisiCalc file format. The VisiCalc rows convert to records, and the columns convert to fields.

- SYLK — Multiplan spreadsheet format in *row major* order. The Multiplan rows convert to records, and the columns convert to fields. dBASE III PLUS will not APPEND FROM a SYLK file saved in *column major* order.

- WKS — Lotus 1-2-3 spreadsheet format, release 1A. The Lotus 1-2-3 rows convert to records, and the columns convert to fields. The file begins with the cell in the upper left-hand corner of the spreadsheet. This is usually cell A1 unless you have renamed it.

Blank rows in any spreadsheet type are converted to blank records in the database file.

When using APPEND FROM to read any of the three supported spreadsheet formats, keep in mind that this command expects the incoming data in a format that matches the database file structure. This means that the spreadsheet should be stored in *row major* (as opposed to *column major*) order, and that column headers should be removed before attempting to read the data into dBASE III PLUS. However, APPEND FROM will store row names as long as the database file structure is designed with them in mind. WKS files should not have leading blank rows or columns. With this type of file, you should justify data in the upper left-hand corner before using APPEND FROM.

## Tips

If you are APPENDing FROM a file for which dBASE III PLUS assumes a particular extension and your file does not have an extension, include a period after the filename.

## Special Cases

Use the IMPORT command to convert pfs:File formats to dBASE III PLUS format.

dBASE III PLUS date fields can APPEND FROM character data in date format. Conversely, character fields in the proper format and size can be APPENDed FROM date fields. From text files, dates can only be APPENDed FROM in the form YYYYMMDD, not delimited (where YYYY is the year, MM is the month, and DD is the day).

## Examples

You can append records from the Travel database file to the end of the Clients database file. (Use a copy of the Clients database file so you can keep the original file intact to use with other examples.)

```
. USE Clients
. APPEND FROM Travel
     49 records added
```

To see the number of records now in the Clients database file, type

```
. ? RECCOUNT()
       98
```

Records 50 through 98 were appended to the Clients database file. Only the fields from the Travel file that were also in the Clients file were appended.

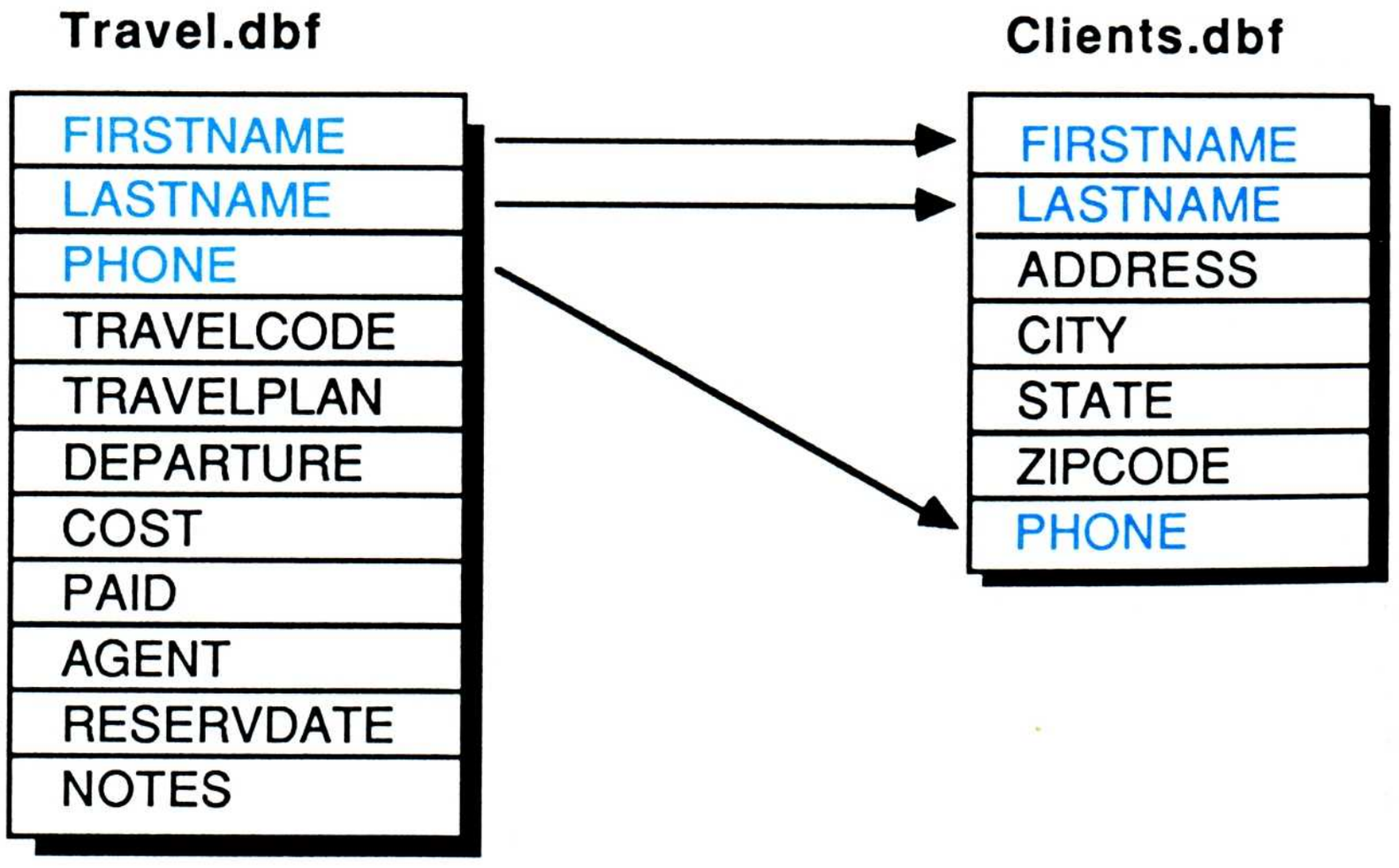



Figure 5-2 The APPEND FROM command

## See Also

IMPORT

## THE ASSISTANT

The Assistant is a full-screen, menu-driven command that prompts you through many dBASE III PLUS commands.

### Syntax

ASSIST

### Usage

The Assistant helps you create new database (.dbf) files and ancillary files: index (.ndx), format (.fmt), label (.lbl), query (.qry), report (.frm), and view (.vue) files. It also allows you to modify these files and perform basic editing, search, and display operations.

Whenever you need help on how to use dBASE III PLUS while in The Assistant, press **F1** for more information on any active main menu option. Press any key to erase the information from the screen and continue.

Each menu option activates a specific dBASE III PLUS command. For more information on a specific command, see the individual entry in the alphabetical listing of commands.

As you choose options from The Assistant, the actual command as you would type it from the dot prompt is built on the action line. In this way, you can learn the dBASE III PLUS command syntax.

### Options

The Assistant menu bar offers eight action choices: **Set Up, Create, Update, Position, Retrieve, Organize, Modify,** and **Tools.**

#### Set Up

The Set Up Menu lets you open files. An option on this menu also lets you exit from the dBASE III PLUS program.

When you choose an option, a list of available drives appears, with the current one as the default. Choose the drive containing your files before choosing a file.

The **Database file** option lets you open a .dbf file. After you choose a drive and database file from the pull-down menu, the next prompt asks if the file is indexed. If it is indexed, and you want to open one or more indexes, enter Y. A list of index files appears. The first one that you select, regardless of its position in the index list, becomes the controlling index. The word **Master** appears next to it. If you select additional index files, they also are opened, and the numbers **02**, **03**, and so on, appear next to them. You can select up to seven index files.

Refer to the USE, INDEX, and SET INDEX commands for more information about opening database and index files.

Individual files cannot be unselected. To save your choices, close the menu with ← or →.

The **Format for Screen** option lets you select a format (.fmt) file from a list of .fmt files, if any exist. Refer to the SET FORMAT TO command for more information on setting up format files.

The **Query** option lets you select a query (.qry) file from a list of .qry files, if any exist. Refer to the CREATE/MODIFY QUERY and SET FILTER TO commands for information on creating and setting up filter (query) files.

The **Catalog** option activates a list of catalog (.cat) files. If you open a catalog, all future selections of files will be from those in the open catalog only. If you do not open a catalog, pull-down selection menus contain all files of the selected type, such as all .dbf or .fmt files, that are on the drive you specify. Refer to the SET CATALOG TO command for information on setting up catalog files.

The **View** option lets you open a view (.vue) file. Refer to the SET VIEW TO command for information on setting up view files.

The **Quit dBASE III PLUS** option closes all open files and returns you to the operating system. It performs the same operation as the QUIT command.

**Create**

The Create Menu lets you create database or other types of dBASE III PLUS files. For each option, you select the drive letter from a submenu, and then enter a filename. If a catalog is open, the file is entered in the catalog. You are prompted to enter a description of the file for the catalog.

The **Database file** option lets you create a .dbf file. After you create the structure of the file, you can choose an option to begin adding data. Refer to the CREATE <newfile>/MODIFY STRUCTURE commands for more information on creating and modifying database files.

The **Format file** option lets you create a custom screen form for data display and entry. The screen form determines how the information is arranged on the screen. Refer to the CREATE/MODIFY SCREEN commands for more information on screen and format files.

The **View** option lets you create a .vue file. View files allow you to work with several .dbf files at the same time. Refer to the CREATE/MODIFY VIEW commands for more information on creating view files.

The **Query** option lets you create a .qry file. This allows you to build a filter condition that limits the information displayed. Refer to the CREATE/MODIFY QUERY commands for more information.

The **Report** option lets you create your own report format (.frm) files. It allows you to output your data in columns, provide headings for each column, and group data to your specification. Refer to the CREATE/MODIFY REPORT commands for more information.

The **Label** option lets you create your own label format (.lbl) files for generating labels. You can specify size of labels, the number of labels on each page, and the information on each label. Refer to the CREATE/MODIFY LABEL commands for more information on creating labels.

### Update

The Update Menu allows you to display and change information in your database files. With this option, you can add new records to an open file, edit existing records, or remove records from the file. When you make changes to a database file, any index (.ndx) files which are open are updated automatically. The Assistant prompts you in most operations by activating menus that let you select field names and conditions of the records you want. For more information on any option's operation from the Update Menu, refer to the commands in this chapter that match the menu option names.

The **Append** option allows you to add records to the end of a file. You can select a custom format file to use with this option or use the one dBASE III PLUS provides.

The **Edit** command allows you to change the contents of a file, record by record. You can select a custom format file with this option or use the one dBASE III PLUS provides.

The **Display** option allows you to view the contents of a single record.

The **Browse** option allows you to display or change the contents of a file, with up to 17 records appearing on the screen at a time.

The **Replace** command allows you to make changes to the contents of a file. With this option, for example, you could change the data in a numeric field for all records in the open database file, or for only those records which match a specified condition.

The **Delete** command marks records for deletion. The records are removed from the file with **Pack**.

The **Recall** command reinstates all or only selected records that were previously marked for deletion.

The **Pack** command permanently removes all records that are marked for deletion from a file.

**Position**

The Position Menu lets you find specified records in your file. Once you locate the record, you can then update or retrieve the data.

The Assistant displays a series of field and condition submenus to help you select records you want to locate.

The **Seek** option is only active if the file is indexed. It allows you to enter an expression that uses the master index key to find the first record in an indexed database file that matches it.

The **Locate** command moves the record pointer to the first record that meets conditions that you specify.

The **Continue** option is only active after a Locate. It moves the record pointer to the next occurrence of the condition specified with Locate.

The **Skip** option moves the record pointer forward or backward a specified number of records.

The **Goto Record** option lets you specify a record number where you want to position the record pointer or go to the top or bottom of the file.

**Retrieve**

These options let you view the contents of a database file, send them to the printer in various formats, and perform summary calculations. For more information on any of the Retrieve Menu options, refer to the commands in this chapter that match the menu option names.

The **List** option lets you view all or selected fields and records. It gives you the choice of viewing records on the screen or printing them.

The **Display** option performs the same function as the **List** option, except that it stops after every full screen, and waits for you to press any key to continue.

The **Report** option lets you display or print a report created with the **Report** option in the Create Menu.

The **Label** option lets you display or print labels created with the **Label** option in the Create Menu.

The **Sum** command lets you total the values of numeric fields for all records, or only selected ones.

The **Average** command lets you average the values of numeric fields for all or selected records.

The **Count** command lets you tally the number of records that meet a specified condition.

**Organize**

The Organize Menu helps you arrange the order of records in your files. Each of the three options in this menu — **Index**, **Sort**, and **Copy** — creates new files. If a catalog is open, the new file is entered in the catalog. The Assistant prompts you to enter a description of the new file for the catalog.

The **Index** option creates an index file. When you select this option, The Assistant prompts you to enter an index key expression on which the database file will be indexed. As you are building an index expression, you can press **F10** to choose fields from a submenu that you want to include in the index expression. Immediately after the index file is created, it is put in USE. Whenever this file is open, the database file from which it was created appears in the order specified, such as alphabetical or chronological order.

The **Sort** command creates a new database file that duplicates all or selected records and fields of the database file in use. In the new sorted file, however, the records are arranged according to the conditions you specified, for example, alphabetical or chronological order. To use this file, go to the Set Up Menu and choose it.

The **Copy** command makes a copy of all or selected portions of the current file. If the file being copied is indexed, the new file will be arranged in the same indexed order, as if it were sorted. If you are working on a hard-disk system, you can use this command to make a copy of a file to a disk on the floppy drive. To use this file, go to the Set Up Menu and choose it.

### Modify

The Modify Menu has the same menu options as the Create Menu. The only difference between the two menus is that the Modify Menu options activate a pull-down menu list of existing files, whereas Create Menu options always prompt you to enter a new filename. For information on each of the options, refer to the CREATE/MODIFY commands in this chapter that match the menu option names.

If a catalog is open, the pull-down filename menu for Modify Menu options only contains files existing in the current catalog. Also, if a catalog is open, only the files that pertain to a particular database file in use are displayed.

If no catalog is open, the pull-down menu filename list contains all of the files of the type selected by a Modify option, for example, all .dbf files or all report (.fmt) files.

**Tools**

The Tools Menu options perform mostly operating system functions. This menu also has **Import** and **Export** options to let you convert dBASE III PLUS files to pfs:File and vice-versa.

The **Set Drive** option changes the default drive. Thereafter, unless you specify differently, dBASE III PLUS searches for files on, and writes any new files you create to, the default drive. Refer to the SET DEFAULT command for more information.

The **Copy File** option makes a duplicate of any type of file (except the database file currently in use), unlike the **Copy** option in the Organize Menu, which only copies .dbf files. Refer to the COPY FILE command for more information.

The **Directory** option allows you to list all files, or all files of a specified type, on the selected drive. Refer to the DIR command for more information.

The **List Structure** option displays the file structure of the open database file and allows you to print the file structure. Refer to the LIST STRUCTURE command for more information.

The **Rename** option lets you give a new name to any file that is not currently open. See the RENAME command for more information.

The **Erase** option permanently removes a file from the directory. Refer to the ERASE command for more information.

The **Import** option converts a pfs:File file to a dBASE III PLUS database (.dbf) file and creates a format (.fmt) file from the pfs:File form. It also creates a view (.vue) file which specifies the database file and the format file created from the pfs:File form. Refer to the IMPORT command for more information.

The **Export** option converts a dBASE III PLUS database (.dbf) file to pfs:File format. If a dBASE III PLUS format (.fmt) file is in use before the export, the format file is also converted to a PFS:file screen form. Otherwise, the pfs:File screen form is created using the default screen form dBASE III PLUS provides. Refer to the EXPORT command for more information.

## AVERAGE

AVERAGE computes the arithmetic mean of numeric expressions.

### Syntax

AVERAGE [ < expression list > ] [ < scope > ] [WHILE < condition > ] [FOR < condition > ] [TO < memvar list > ]

### Defaults

All records are averaged unless otherwise specified by the scope or the WHILE or FOR clause. All numeric fields are averaged unless otherwise specified by an expression list. The result of AVERAGE is displayed on the screen as long as SET TALK is ON.

### Examples

To obtain the average cost of the sales for agent JT in the Travel database file:

```
. USE Travel
. AVERAGE Cost FOR Agent = 'JT'
     16 records averaged
    Cost
1553.94
```

To find the average costs and commission rates for Travelplans in Tours.dbf and store the results to memory variables Aveprice and Avecomm:

```
. USE Tours
. AVERAGE Unitcost, Percent TO aveprice, avecomm
     30 records averaged
 Unitcost  Percent
  1542.10    7.302
. ? aveprice, avecomm
     1542.10          7.302
```

### See Also

SET HEADING, SET TALK, SUM

## BROWSE

BROWSE is a full-screen, menu-assisted command for editing and appending records in database (.dbf) files. It presents up to 17 records at once and as many fields as fit across the screen.

## Syntax

BROWSE [FIELDS < field list > ] [LOCK < expN > ] [FREEZE < field > ] [NOFOLLOW] [NOMENU] [WIDTH < expN > ] [NOAPPEND]

## Usage

The BROWSE menu bar is activated when you press **F10**. Once you choose an option, the menu bar disappears. You can set many of the same options provided on the menu bar by entering them at the dot prompt (see the Command Line Options below).

To turn the navigation key menu on and off, press **F1**.

## Command Line Options

**FIELDS**

You specify a fields list to indicate the fields you want to display, and the order in which you want them to appear.

**LOCK**

This specifies the number of fields that you want to keep displayed on the screen, starting from the first field in the fields list. When you press **Ctrl-→** or **Ctrl-←** to scroll fields left or right, the number of fields you *locked* will remain displayed in the same position on the screen.

**FREEZE**

Freezing a field prevents you from editing any of the other fields in the field list until you reissue the BROWSE command. Although you can edit only one field, all the other fields are still displayed.

**NOFOLLOW** The **NOFOLLOW** option only has an effect on indexed files. Ordinarily, if you alter the contents of the key field of a record, the record pointer is repositioned immediately according to the new contents of the field. The edited record remains the current record. If you issue **NOFOLLOW** before changing the field's contents, the record is repositioned, but the record that took its original place in the file order becomes the current record.

**NOMENU** The **NOMENU** option suppresses display of the menu bar.

**WIDTH** The **WIDTH** option limits the number of character positions (between 4 and 99) that BROWSE may display for any character field. You can move (scroll) within fields using the ← and → keys, and move from field to field using the **Home** and **End** keys.

**NOAPPEND** The **NOAPPEND** option prevents you from adding records to the current file when in BROWSE.

## Menu Bar Options

The Browse Menu bar offers up to six selections: **Top, Bottom, Record No., Lock, Freeze,** and **Find.**

**Bottom** The **Bottom** option moves the record pointer to the last record in the file, making it the current record.

**Top** The **Top** option moves the record pointer to the first record in the file, making it the current record. The first record may not be record number 1 if the file is in an indexed order.

**Lock** The **Lock** option allows you to specify a number of columns (fields) that you want to remain in the same position on the screen, while other fields scroll horizontally across the display. You are prompted to enter the number of columns (fields) currently displayed on your screen that you want to remain.

**Record No.** The **Record No.** option allows you to go to a specific record number. You are prompted to enter a record number, after which the record pointer moves to highlight the record you selected.

**Freeze** Choosing **Freeze** prompts you to enter the name of the field you want to edit. All other fields will remain displayed, but cannot be edited. See the explanation under Command Line Options.

**Find** The **Find** option appears on the menu bar only if the database file in use is indexed. You are prompted to enter a value based on the index key that identifies the record you want to locate.

## See Also

APPEND, EDIT

## CALL

The CALL command allows you to call binary file program modules loaded in memory from within a dBASE program file. You must have first loaded the binary program files in memory using the LOAD command.

## Syntax

CALL <module name> [WITH <expC> / <memvar>]

## Usage

The CALL command executes a binary program file that has been loaded in memory with the LOAD command. dBASE III PLUS treats each loaded file as a subroutine or module rather than as an external program (which could be executed with the RUN command). As a result, each time you want to execute the program, it can simply run from memory without having to be reloaded from a disk. Up to 16 binary program files can be loaded in memory at one time, and each can be up to 32,000 bytes in length.

When you CALL the binary program file, you specify the name of the file without the .bin extension. The program module name is the same as the file loaded in memory. When you call the file, you can pass either a character expression or a memory variable of any data type to the binary program file. All character type memory variables and expressions end with a null (ASCII value of zero).

When the program module is called, the Code Segment (CS) points in memory to the beginning of the module. The Data Segment (DS) and BX registers contain the address of the first byte of the memory variable or character expression.

## Options

The WITH option is a character expression or a memory variable. Memory variables can be of any type.

## Example

Refer to the LOAD command for an example of the CALL command calling a LOADed binary program file. The CALL passes a character string, which is used to change the shape and size of the cursor.

## See Also

LOAD, RELEASE

## CANCEL

CANCEL stops the execution of a command file, closes all open command files, and returns dBASE III PLUS to the dot prompt. CANCEL does not close procedure files.

### Syntax

CANCEL

### Usage

dBASE III PLUS ignores any text on lines in a dBASE program (.prg) file following CANCEL.

### Example

To stop command processing when an error is found, include CANCEL within an IF...ENDIF or DO CASE...ENDCASE structure. In the following example, the Error memory variable was previously established:

```
IF Error                && If error is true,
    CANCEL              && command file execution
ENDIF                   && is cancelled.
```

### See Also

RETURN

## CHANGE

The CHANGE command provides full-screen editing of specified fields and records in the active database file.

### Syntax

CHANGE [ < scope > ] [FIELDS < field list > ]
[WHILE < condition > ] [FOR < condition > ]

### Defaults

If CHANGE is used without parameters, the current record is displayed.

### Usage

The CHANGE command is identical to the EDIT command. CHANGE uses the same full-screen cursor control keys as both EDIT and MODIFY COMMAND. (Refer to Appendix B.) The arrow keys move the cursor within a record. **PgUp** backs up to the previous record, or to previous fields of the same record if not all the fields fit on one screen. Similarly, **PgDn** advances to the next record, or to additional fields of the same record if all the fields do not fit on one screen. **Ctrl-End** exits CHANGE and saves changes to the current record. **Esc** exits and saves all changes except those made to the current record.

To edit a memo field, place the cursor on the word **memo** and press **Ctrl-PgDn**. To finish editing the memo field, press **Ctrl-End** to save or **Esc** to ignore changes. Use the same cursor control keys as with MODIFY COMMAND.

If you use the CHANGE command at the dot prompt, control automatically returns to the dot prompt after you display (CHANGE) the last record that meets the specified condition.

### Options

All records are edited sequentially unless otherwise specified by the scope or the WHILE or FOR clause. All fields are displayed unless you limit the fields with a FIELDS clause.

### Example

The Tours database file on the Sample Programs and Utilities disk is used for the following example.

To edit the fields Departure, Unitcost, and Seats_open for all database records:

```
. SET STATUS ON
. USE Tours
. CHANGE FIELDS Departure, Unitcost, Seats_open
```

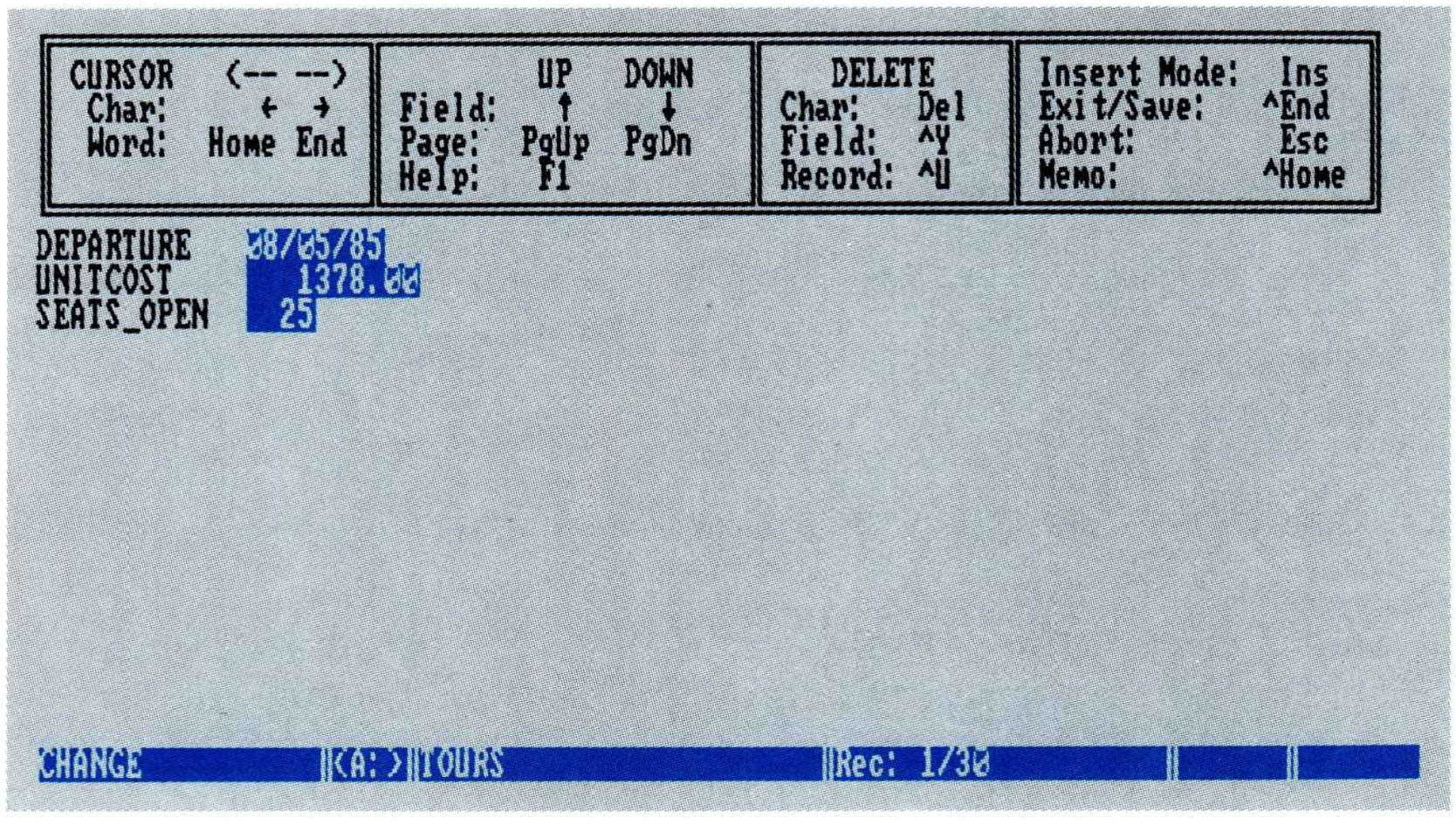



Figure 5-3 The CHANGE command

## See Also

EDIT, MODIFY COMMAND, SET FORMAT TO

## CLEAR

CLEAR erases the screen. If STATUS is ON, CLEAR repositions the cursor to the row just above the status bar (screen coordinates 21, 0) after the screen clears. If STATUS is OFF, CLEAR repositions the cursor to the lower left-hand corner of the screen (coordinates 24, 0) after the screen clears.

### Syntax

CLEAR

### Usage

CLEAR also releases all pending GETs created with the @...GET command that are in the area erased by the CLEAR command.

### See Also

@, CLEAR GETS

## CLEAR ALL

CLEAR ALL closes all open database files, releases all memory variables, and selects work area 1.

### Syntax

CLEAR ALL

### Usage

Closing the database files also closes their associated index (.ndx), format (.fmt), and memo (.dbt) files, and also closes a catalog (.cat) file if one is active.

### See Also

CLEAR MEMORY, CLOSE, RELEASE ALL

## CLEAR FIELDS

CLEAR FIELDS releases the fields list created by the SET FIELDS TO command.

### Syntax

CLEAR FIELDS

### Usage

CLEAR FIELDS has no effect unless a SET FIELDS TO command has been given either directly or in a view.

Unlike SET FIELDS TO, which removes only the active database fields from the fields list, CLEAR FIELDS releases all fields, including those set in other work areas.

CLEAR FIELDS automatically performs a SET FIELDS OFF.

### See Also

SET FIELDS, SET VIEW

## CLEAR GETS

CLEAR GETS releases all @...GETs issued since the last CLEAR ALL, CLEAR GETS, or READ command.

### Syntax

CLEAR GETS

### Usage

Unless the GETS parameter is redefined using a Config.db file, dBASE III PLUS permits 128 @...GETs before a CLEAR GETS must be issued.

The maximum number of GETs allowed is 1,023. However, the number you can actually use is limited by the amount of memory in your computer. (See *Getting Started* for memory limitations.) For most purposes, the default value of 128 is sufficient.

CLEAR GETS does not release memory variables.

## CLEAR MEMORY

CLEAR MEMORY releases all memory variables.

### Syntax

CLEAR MEMORY

### Usage

CLEAR MEMORY performs the same function as the RELEASE ALL command when issued at the dot prompt. When used in programs, however, CLEAR MEMORY releases all PUBLIC and PRIVATE memory variables, while RELEASE ALL only releases PRIVATE memory variables declared in the program currently being executed.

# CLEAR TYPEAHEAD



## CLEAR TYPEAHEAD

CLEAR TYPEAHEAD empties the type-ahead buffer.

## Syntax

CLEAR TYPEAHEAD

## Usage

Use CLEAR TYPEAHEAD when you want to make sure that there is no text in the type-ahead buffer. This is particularly useful in programs when you want the user to enter information before the program continues. For instance, place it before CHANGE, READ, WAIT, or similar commands to ensure that dBASE III PLUS acts upon the correct characters.

## See Also

SET TYPEAHEAD TO

## CLOSE

CLOSE is used to close alternate files, database files, format files, index files, and procedure files.

## Syntax

CLOSE < file type > /ALL

CLOSE ALL closes all files of all types without affecting memory.

To CLOSE a particular file type, follow the command with one of the following keywords:

ALTERNATE
DATABASES
FORMAT
INDEX
PROCEDURE

If you want to close only the currently selected database file and its associated files, issue the USE command, rather than the CLOSE command.

CLOSE DATABASES does not affect work area 10 if a catalog is open.

## Examples

To close all open files without affecting memory:

```
. CLOSE ALL
```

To close all open database, index, and format files:

## CONTINUE

CONTINUE searches for the next record in the active database file that meets the condition specified by the most recent LOCATE command.

### Syntax

CONTINUE

### Usage

A CONTINUE search ends when it finds a record that meets the specified LOCATE condition or when it reaches the end of the LOCATE *scope*.

LOCATE and CONTINUE are specific to the work area in which you issue them. You can issue different LOCATE/CONTINUE commands in each work area. If you leave a work area, the LOCATE/CONTINUE condition will still be in effect when you return.

### Record Pointer

When a CONTINUE is successful, the record number is displayed. Otherwise dBASE III PLUS displays the message **End of LOCATE scope**. If the search is unsuccessful, the record pointer is positioned at the last record of the LOCATE scope. If this is the last record in the database, EOF() will return a True value.

### Example

To locate records containing CN7 or AV10 in the Travelcode field in the Travel database file:

```
. USE Travel
. LOCATE FOR Travelcode = 'CN7' .OR. Travelcode = 'AV10'
Record =       2
. CONTINUE
Record =      15
. CONTINUE
Record =      47
. CONTINUE
End of LOCATE scope
```

The message **End of LOCATE scope** indicates that no more records fit the FOR condition.

### See Also

EOF(), FOUND(), LOCATE

## COPY

COPY duplicates all or part of an active database file to a new file. COPY is the primary command used to export data to non-dBASE III PLUS programs.

## Syntax

COPY TO <new file> [<scope>] [FIELDS <fields list>] [FOR <condition>] [WHILE <condition>] [[TYPE] <file type>]

## Defaults

All records are copied, including records marked for deletion, unless SET DELETED is ON, or you specify otherwise with a scope, FOR, or WHILE clause. All fields are copied unless you specify a FIELDS list or use the SET FIELDS TO command. Memo fields are copied only if the new file is another dBASE III PLUS database file.

The directory listing of new files created by the COPY command shows the dates on which the new files were created.

If a file type is not entered with the command, the file is copied to another dBASE III PLUS database (.dbf) file.

## Usage

If the TO file is another dBASE III PLUS database (.dbf) file, do *not* specify TYPE or [<file type>]. Otherwise you need to specify one of the file type options supported, to transfer the dBASE III PLUS database file to another software program or create a delimited text file.

**NOTE**
dBASE III PLUS may allow you to build files with more fields, more characters per field, or more records than certain other software programs allow. In such cases, when you copy a dBASE III PLUS file, original data may be lost. Note the limitations of the software program to which you're exporting a file before creating the files with COPY.

## Options

The options for < file type > are:

- DELIMITED [WITH BLANK/ < delimiter > ] — Delimited format ASCII text (.txt) file. Data from fields is copied character by character beginning with the first character on the left. Each record ends with a carriage return and line feed. All fields are separated by commas unless you specify DELIMITED WITH BLANK. In addition, character fields are enclosed by double quotes (″), unless you specify a different delimiter character using DELIMITED WITH < delimiter > or you specify DELIMITED WITH BLANK.

  DELIMITED WITH BLANK separates each field with a single space character instead of a comma. Character fields are treated like any other field and are not enclosed by delimiter characters.

  DELIMITED WITH < delimiter > allows you to enclose character fields with a delimiter character of your choice.

- SDF — System Data Format ASCII file. Data in fields is copied character by character beginning with the first character on the left. Each record has the same length and ends with a carriage return and line feed.

- DIF — VisiCalc file format. Database records are converted to VisiCalc rows, and database fields are converted to VisiCalc columns.

- SYLK — Multiplan spreadsheet formula. Database records are converted to Multiplan rows, and database fields are converted to Multiplan columns.

- WKS — Lotus 1-2-3 spreadsheet format, release 1A. Database records are converted to Lotus 1-2-3 rows, and database fields are converted to Lotus 1-2-3 columns.

  When using COPY to write any of the three supported spreadsheet formats, the field names are written as column headers in the resulting file. The file is created in row major order.

## Special Cases

Use the EXPORT command to convert dBASE III PLUS files to pfs:File format.

## Tips

Do not use the single letters A through J or M as a database filename if you COPY TO a dBASE III PLUS database file. However, you can, for example, specify AA as a database filename.

## Example

Clients.dbf contains the fields Firstname, Lastname, Address, City, State, Zipcode, and Phone. You can create a new file that contains only the fields Firstname, Lastname, and Phone for all California residents:

```
. USE Clients
. COPY TO Calif FIELDS Firstname,Lastname,Phone FOR State = 'CA'
     34 records copied
. USE Calif
. LIST
Record#  FIRSTNAME       LASTNAME        PHONE
      1  Claire          Buckman         (555)456-9059
      2  Rick            Lisbonn         (555)455-3344
      3  Lisa            Kafmanan        (555)860-0300
      4  Jay             Johnson         (555)860-6784
      5  Rick            Zambini         (555)345-1945
      6  Kim             Arlich          (555)624-8773
      7  John            Montovan        (555)566-3403
         .               .               .
         .               .               .
         .               .               .
     34  Sandy           Young           (555)574-0777
```

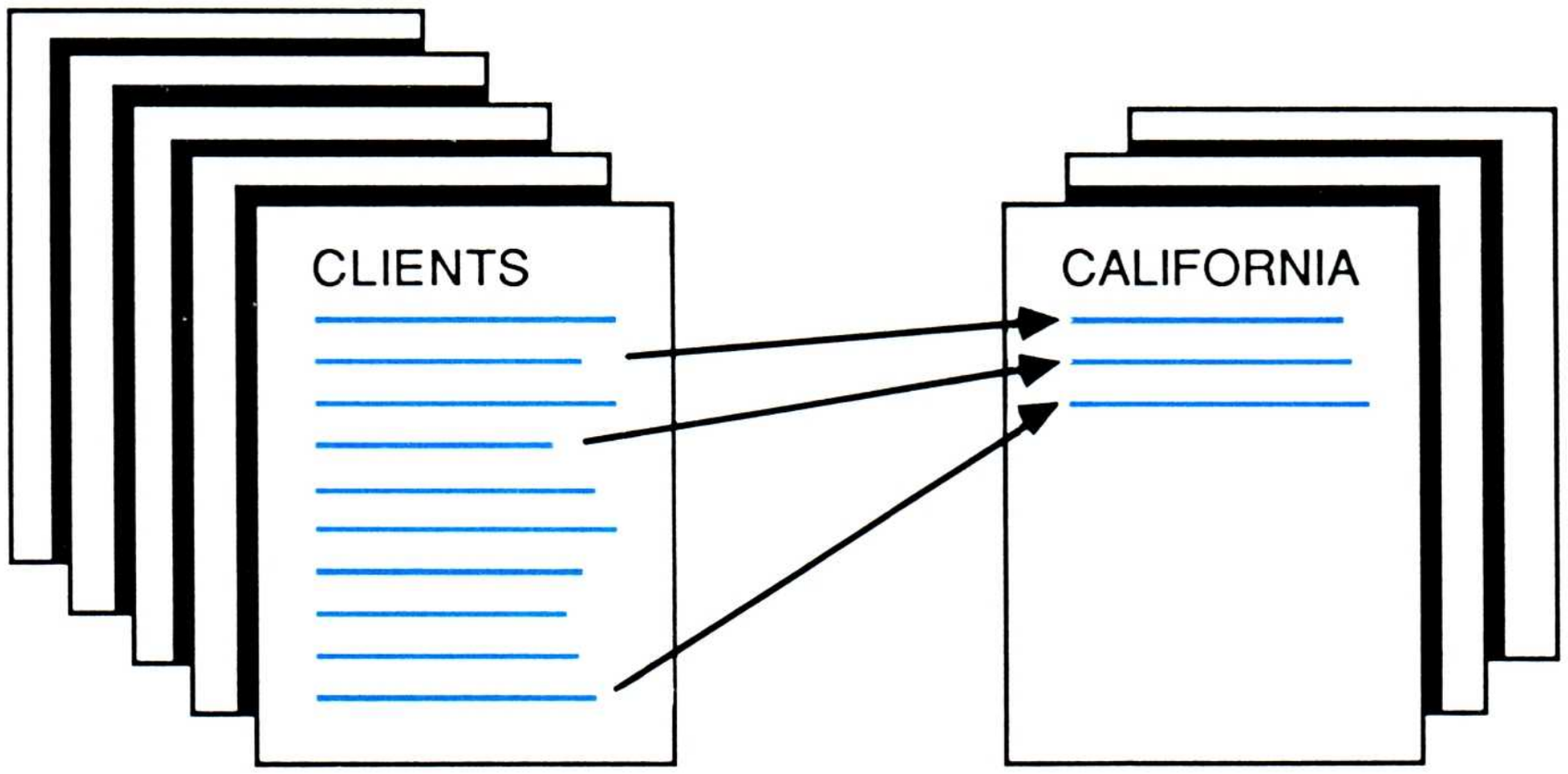



Figure 5-4 The COPY command

## See Also

APPEND FROM, COPY FILE, COPY STRUCTURE, EXPORT, SET DELETED, SET SAFETY

## COPY FILE

COPY FILE creates a duplicate of any closed file.

### Syntax

COPY FILE <file1> TO <file2>

### Usage

The first file is copied to the second file. Both filenames you specify must include the file extension and the drive (if you want to copy a file to or from a drive which is not the default). However, you do not need to specify the drive location of the first file if it is specified in a dBASE III PLUS path with the SET PATH command (or PATH = in the dBASE III PLUS Config.db file). You cannot use COPY FILE to copy an open file. If you copy a database file that has memo fields, you must copy the associated memo (.dbt) file separately.

### Tips

Do not use the single letters A through J or M as database filenames because they are reserved for ALIAS names. However, you can, for example, specify AA as a database filename.

### Example

To make a duplicate of a dBASE program file:

```
. COPY FILE Cbmenu.prg TO Checking.prg
5120 bytes copied
```

### See Also

COPY

## COPY STRUCTURE

COPY STRUCTURE copies only the structure of the active database file to a new file.

### Syntax

COPY STRUCTURE TO <filename> [FIELDS <field list>]

### Defaults

The filename must include the drive designator if the file is not on the default drive. Unless otherwise specified, the filename of the TO file is assigned an extension of .dbf.

### Usage

If you COPY a database file, the entire structure is copied unless limited by the FIELDS option.

### Example

```
. USE Travel
. COPY STRUCTURE TO Temp
```

### See Also

DISPLAY STRUCTURE, SET SAFETY

# COPY STRUCTURE EXTENDED

## COPY STRUCTURE EXTENDED

COPY STRUCTURE EXTENDED creates a new database file with four fields: FIELD_NAME, FIELD_TYPE, FIELD_LEN, and FIELD_DEC. The contents of the new file consist of the field names, data types, field widths, and number of decimal places (in a numeric field) for all of the fields in the currently active database file.

## Syntax

COPY TO <new file> STRUCTURE EXTENDED

## Usage

This command is most often used within application programs to create or modify a database file without using the interactive CREATE and MODIFY STRUCTURE commands. The CREATE FROM command is then used to create a new database file from the STRUCTURE EXTENDED file.

## Example

To copy the field information from the Clients database file into the STRUCTURE EXTENDED file Newnames:

```
. USE Clients
. DISPLAY STRUCTURE
Structure for database : B:Clients.dbf
Number of data records :       49
Date of last update    : 07/24/85
Field  Field Name  Type          Width     Dec
    1  FIRSTNAME   Character        20
    2  LASTNAME    Character        20
    3  ADDRESS     Character        25
    4  CITY        Character        20
    5  STATE       Character         2
    6  ZIPCODE     Character         5
    7  PHONE       Character        13
** Total **                       106
```

*(Example continued on next page)*

```
. COPY TO Newnames STRUCTURE EXTENDED
. USE Newnames
. DISPLAY STRUCTURE
Structure for database : B:Newnames.dbf
Number of data records :          7
Date of last update    :  09/20/85
Field  Field name  Type          Width    Dec
    1  FIELD_NAME  Character        10
    2  FIELD_TYPE  Character         1
    3  FIELD_LEN   Numeric           3
    4  FIELD_DEC   Numeric           3
** Total **                        18
. LIST
Record#  FIELD_NAME FIELD_TYPE  FIELD_LEN FIELD_DEC
      1  FIRSTNAME  C                  20         0
      2  LASTNAME   C                  20         0
      3  ADDRESS    C                  25         0
      4  CITY       C                  20         0
      5  STATE      C                   2         0
      6  ZIPCODE    C                   5         0
      7  PHONE      C                  13         0
```

## See Also

CREATE FROM

## COUNT

COUNT tallies the number of records in the active database that match specified conditions.

## Syntax

COUNT [ < scope > ] [WHILE < condition > ] [FOR < condition > ] [TO < memvar > ]

## Defaults

Unless otherwise specified by the scope, FOR, or WHILE options, all database records are COUNTed.

## Example

The Clients database file is used in the following example.

To determine the number of clients in the Clients database file from the states of NV or AZ, and save the result in the Mstate memory variable:

```
. USE Clients
. COUNT FOR State = 'NV' .OR. State = 'AZ' TO mstate
     10 records
. ? mstate
          10
```

# CREATE < newfile >

**CREATE** The CREATE command allows you to create a new database file structure and add new records to the new database file.

**Syntax** CREATE < newfile >

**Usage** See CREATE < newfile > /MODIFY STRUCTURE

> **NOTE**
> You use MODIFY STRUCTURE to modify the structure of an existing database file structure.

# CREATE FROM

## CREATE FROM

CREATE FROM forms a new database file from the structure created with the COPY STRUCTURE EXTENDED command.

## Syntax

CREATE <new file> FROM <structure extended file>

## Usage

The file created with CREATE FROM becomes the active database file. If the CREATE FROM operation fails for any reason, no database file remains selected in the current work area.

Do not use the letters A through J or M as database filenames, because they are reserved for ALIAS names. However, you can, for example, use AA as a filename.

## Example

First follow the example provided for the COPY STRUCTURE EXTENDED command, in which you create the Newnames.dbf file from Clients.dbf. Now, using the CREATE FROM command, you can create a new database file with the same structure as Clients.dbf:

```
. CREATE Nclients FROM Newnames
. DISPLAY STRUCTURE
Structure for database: B:Nclients.dbf
Number of data records:       0
Date of last update    : 08/02/85
Field  Field Name  Type       Width    Dec
    1  FIRSTNAME   Character     20
    2  LASTNAME    Character     20
    3  ADDRESS     Character     25
    4  CITY        Character     20
    5  STATE       Character      2
    6  ZIPCODE     Character      5
    7  PHONE       Character     13
** Total **                     106
```

## See Also

COPY STRUCTURE EXTENDED

# CREATE/MODIFY LABEL

## CREATE/ MODIFY LABEL

CREATE/MODIFY LABEL is a full-screen, menu-driven command which allows you to create label form (.lbl) files using the fields specified in the current database file or in other related database files.

## Syntax

CREATE/MODIFY LABEL <filename> /?

## Defaults

Unless you specify otherwise, dBASE III PLUS creates the file with the .lbl extension. If a catalog is open and SET CATALOG is ON, the label file is added to the catalog.

## Usage

You use the CREATE LABEL command to create a new label form file. You specify the size of the label you want to create and the number of printed lines on each label. These values should match the label forms or paper on which you want to print.

You use MODIFY LABEL to modify an existing label form file. If you use the catalog query (?) with the MODIFY LABEL command, you will be prompted to select a file from the list of label form files already in the catalog.

Using CREATE or MODIFY LABEL, you specify the information you want on each label. For each line on the label, you specify a dBASE expression, which may combine character strings, fields, and functions. The fields you specify in the expression must be from the current database file or in other related database files (as in a view file).

After you finish creating the label form and saving the label file, you can display or print labels with the LABEL command. You can use a filter condition or filter query (.qry) file to specify particular records you want to appear on labels.

While you are creating the format for labels, you can toggle a cursor navigation menu on or off the screen by pressing **F1**. This menu describes the operations you can perform, and the keys you can use.

## Options

The CREATE/MODIFY LABEL menu bar displays three choices: **Options**, **Contents**, and **Exit**.

### Options

The Options Menu allows you to determine the size and number of lines printed on labels.

The **Predefined size** option lets you choose one of five standard label sizes:

| Width | Height | Number of labels across |
|---|---|---|
| 3-1/2" | 15/16" | 1 |
| 3-1/2" | 15/16" | 2 |
| 3-1/2" | 15/16" | 3 |
| 4" | 1-7/16" | 1 |
| 3-2/10" | 11/12" | 3 (Cheshire style) |

Table 5-1 Standard label sizes

When you choose a standard label size, dBASE III PLUS automatically fills in values for the remaining options. Otherwise, you can adjust these values to suit your needs, entering a new value or using the ↑ or ↓ arrow keys to change the values. The settings you enter should match the label forms or paper on which you are printing.

Each of the options has a set range of allowable values.

| Option | Range |
|---|---|
| Label width | 1-120 characters |
| Label height | 1-16 lines |
| Left margin (offset by SET MARGIN TO setting) | 0-120 characters |
| Lines between labels | 0-16 lines |
| Spaces between labels | 0-120 characters |
| Labels across page | 1-5 labels. |

Table 5-2 Option Menu range specifications

The maximum width of all labels is 250 characters.

The LEFT margin option is added to the current setting of the SET MARGIN TO command. For example, if you enter a left margin of 5, and SET MARGIN was previously set to 11, your labels will print with a left margin of 16.

**Contents**

You design the layout of the label through the Contents Menu. You can enter as many lines as you specified in the Options Menu. For each line on the label, specify any valid dBASE expression, which may combine character strings, fields, and functions. The fields you specify on the label form must be from the current database file or from other related files.

Press ↵ to begin entering fields or an expression on each line in the Contents Menu. When you're entering more than one field on a line, you need to separate the fields with a comma or plus sign. If you use a comma, dBASE III PLUS removes trailing blank spaces in a field and then inserts one blank space before the next field or character on the label. A plus sign does not remove trailing blanks.

As you enter an expression and reach the end of an entry line in the Contents Menu, characters in the expression scroll horizontally. You use the ← and → keys to move the cursor back and forth on the line. If you want to see the entire line all at once, you can press **Ctrl-PgDn** to make your entry on a full-length editing line at the bottom of the screen. When you're finished with an entry on this line, press **Ctrl-PgUp** to return to the Contents Menu. You can use this *zoom* option to create or edit expressions for each line of the label form.

Press **F1 Help** if you're not sure which keys you can use while in the Contents Menu. While you're entering an expression, you can also press **F10** to display a list of field names from the database file in use or fields specified by a view file. You select a field by highlighting it on the field list and pressing ↵. The field automatically appears in the label expression you're entering. You can use this feature as a shortcut to specifying fields, or as a reminder of the fields in the file.

When creating a label, you can use functions in an expression to combine fields and character strings on a line. For example, to put a comma on a line without having a space between a field and the comma (as in Los Angeles, CA 90049), you could enter the following expression:

```
Label contents 1:TRIM(City) + ',',State,Zipcode
```

The TRIM( ) function removes *all* trailing blanks, and the plus sign puts the comma immediately after City. The commas before and after State insert blank spaces.

**Exit**

The **Save** option saves the label file. The **Abandon** option abandons the creation of a new label file or changes made to an existing one. If you're creating or modifying a label file and want to abandon the file, you can press **Esc**. You are prompted to abandon your work on the label file, or resume it.

## See Also

LABEL, TRIM()

# CREATE/MODIFY QUERY

## CREATE/ MODIFY QUERY

CREATE/MODIFY QUERY creates or modifies a file which specifies conditions that records in a database file must match to be accessed.

## Syntax

CREATE/MODIFY QUERY < .qry filename > /?

## Defaults

Unless otherwise specified, dBASE III PLUS supplies the .qry extension. If a catalog is open and SET CATALOG in ON, the query file is added to the catalog.

## Usage

CREATE QUERY and MODIFY QUERY are full-screen, menu-driven commands that create or modify a filter (query) file associated with a database or view file. A query file allows only the records that meet the specified conditions to be displayed when commands are executed after the query file is set up.

You use CREATE QUERY to build a new query file. You use MODIFY QUERY to modify an existing query file. If you want to modify an existing query file, and a catalog file is active, you can use MODIFY QUERY ?. This displays a list of .qry files associated with the currently selected database file or view files.

After you create a query file and exit with the **Save** option, dBASE III PLUS automatically activates the new or modified query file. To deactivate the filter, use SET FILTER TO ↵. To activate an existing query file, use the SET FILTER TO FILE < .qry > /? command.

You can specify up to seven separate conditions in a query file. Each condition specifies a test for fields in the database file(s) for which the query file is constructed.

When you are constructing a query file, you first enter a field name. Then you specify an operator specific to the data type of the field name. Finally, you specify the condition that the field needs to match. If you want to add additional conditions, specify a **Connect** operator — .AND., .OR., .AND. .NOT., or .OR. .NOT. When you're all finished, select **No combination**.

An example of a condition which you can construct in a query file is:

TRAVELCODE = 'CI10' .AND. PAID Is False

## Options

dBASE III PLUS provides menus for building the conditions in the query file. The menu selections are **Set Filter**, **Nest**, **Display**, and **Exit**.

### Set Filter

A table appears with this menu which displays existing filter conditions. You need to make an entry for each option in this menu. The highlight automatically moves to the next option as you complete entries.

The **Field Name** option offers a submenu of field names to choose from.

The **Operator** option activates a list of operators specific to the data type of the field specified in **Field Name**.

The **Constant/Expression** option is the only one in which you can make an entry rather than select one from a list of fields or operators. If you want to specify fields from the submenu of field names into the expression you are entering, press **F10**. If you make your own entry, the data type of the expression must match the field name specified. That means you have to enclose character strings in quotes, or include any functions necessary to convert data. Date entries, however, are automatically converted, so you can enter them directly in the format *mm/dd/yy* without using the CTOD( ) function.

The **Connect** option links conditions. The operators linking conditions together are .AND., .OR., .AND. .NOT., and .OR. .NOT. You pick the operator you want to use from a selection menu. At the end of the last condition, select the **No combination** option.

The **Line Number** option lets you select one of the seven conditions so you can edit it. You can change the field name, operator, or expression for the selected line in the query table.

- To insert a new condition on the line at the current line number, press **Ctrl**-N.
- To delete the condition on the current line, press **Ctrl**-U.

### Nest

Nesting allows you to determine the order in which conditions are evaluated by adding or removing parentheses around conditions.

**Start** options add or remove a left parenthesis at the beginning of a line on the conditions table. After choosing **Start**, enter the line number.

**End** options add or remove a right parenthesis after the Constant/Expression on the conditions table. After choosing **End**, enter the line number.

**Display**

This allows you to view individual records to see if you set the filter properly. It displays records which match the conditions set up in the query one at a time. You can view additional records, or view remaining fields of the same record, by pressing **PgUp** or **PgDn**.

**Exit**

The **Save** option saves a new or modified query file. This automatically activates the .qry file.

The **Abandon** option abandons the creation or modification of the query file. You can also leave in the middle of creating or modifying a query file by pressing **Esc**.

## See Also

SET CATALOG, SET FILTER

# CREATE/MODIFY REPORT

## CREATE/ MODIFY REPORT

CREATE/MODIFY REPORT is a full-screen, menu-driven command used to create or change report format (.frm) files.

## Syntax

CREATE/MODIFY REPORT < .frm filename > /?

## Defaults

Unless you specify otherwise, report files are given the .frm extension. If a catalog is open and SET CATALOG is ON, the .frm file is added to the catalog.

## Usage

You use the CREATE REPORT command to create a report form (.frm) file, and the MODIFY REPORT command to modify an existing one. If you have a catalog open, you can enter the command MODIFY REPORT ?. This command displays a list of .frm files associated with the active database file, from which you can select.

The report form file specifies the basic arrangement of the report, the location of fields in columns, the text for a report title, and headings for columns. The report form file may specify fields from one database file or fields from related database files. The REPORT command is used to print reports using the report form file in which fields are printed in table format under the respective columns you specified.

You need to press ↵ before you can make entries in the report form menus, to change report settings, or before you specify the headings and fields on the report. **Title** and column **Heading** entries are made in a four-line entry area. Pressing **Ctrl-End** completes your entry.

A menu at the bottom of the screen displays the keys you can use while creating or modifying a report. Press **F1 Help** to display this menu or clear it from the screen.

Entries for report column and grouping expressions are made on the entry lines provided within the menus. As your entries reach the end of the line, the characters are scrolled horizontally to the left to allow you to enter more text. You can use the ← and → keys to move the cursor back and forth on the line, to review or edit your entries.

To see the entire expression as you edit it, press **Ctrl-PgDn** to display a full-length editing line at the bottom of the screen. You can use this *zoom* option to edit each report column expression. When you finish editing these lines, press **Ctrl-PgUp** to return to the editing line in the Groups or Columns Menus.

While you're entering an expression, you can press **F10** to display a list of field names from the database file in use or fields specified by the SET FIELDS TO command. You select a field by highlighting it on the field list and pressing ↵. The field automatically appears in the expression you're entering. You can use this feature as a short-cut to specifying fields, or as a reminder of the fields in your database file.

After you've finished creating a report and have saved it, the REPORT command will display the report on the screen, or print it. Figure 5-5 shows a finished report, using the Traveler report form on the Sample Programs and Utilities disk and a filter selecting records for two trips, with Travelcodes AO26 and PC14. The report includes subtotals for the numeric Cost field in the Travel database file and provides a grand total at the end of the report

```
Page No.    1
06/26/86

                        TRIP RESERVATION REPORT
                        Bon Voyage Travel Agency
                    Trip and Passenger Reservation List

 Last            First           Departure       Package  Paid?
 Name            Name            Date              Price  (Y/N)


** Travel Code A026
 Dickerson       Lori            10/30/85        3295.00  .F.
 Gilbert         Chuck           10/30/85        3295.00  .F.
 ** Subtotal **                                  6590.00
** Travel Code PC14
 Daniels         Dominique       08/15/85        1595.00  .F.
 Rizzo           Ann             08/15/85        1595.00  .F.
** Subtotal **                                   3190.00
*** Total ***                                    9780.00
```

Figure 5-5 A finished report

## Options

The CREATE/MODIFY REPORT menu bar offers five choices: **Options**, **Groups**, **Columns**, **Locate**, and **Exit**.

### Options

The **Page title** option offers four lines in which to type a title for the report. Pressing ↵ at this option opens an editing screen for typing. Pressing **Ctrl-End** closes the entry and returns back to the menu.

The **Page width** option determines the number of characters across a page. The default value is 80 characters. The range of values allowed is 1 to 500.

The **Left margin** option determines the number of spaces the report is offset on the left side of the page. You can specify an additional offset using the SET MARGIN command.

The **Right margin** option determines the number of spaces the report is offset from the right side of the page.

dBASE III PLUS checks the page width and the left and right margin entries for consistency.

The **Lines per page** option lets you put between 1 and 500 lines on a page before the page EJECTs. It does not reset the default page length on your printer.

The **Double space report** option lets you choose between single- or double-spaced printing. The default value is NO, which creates a single-spaced report.

The **Page eject before printing** option has a default of YES, which causes the printer to automatically eject a page before starting to print. If you change it to NO, the report starts printing wherever the paper is aligned in the printer.

You can also suppress the page eject by including a NOEJECT when you issue the REPORT command to send the report to the screen or printer. This overrides a YES on the Options Menu.

The **Page eject after printing** option has a default value of NO. You can eject the last page of the report after printing the report by changing the value to YES. This is equivalent to issuing EJECT after printing a report.

**NOTE**
If you want to print several short reports on one page, do not change the default of No for **Page eject after printing**.

The **Plain page** option determines whether the page number, system date, and any heading defined with the **HEADING** option of the REPORT command are included on each page of the report. With the default value, NO, these pieces of information are included on each page. If the **Plain page** setting is YES, the title appears only on the first page, and no heading, page number, or date appears at all.

You can also activate the **Plain page** option when you issue the REPORT command to output the report to the screen or printer. If you include PLAIN with the REPORT command, it overrides a NO set in the Options menu; page numbers, dates, and titles are not included in the report.

**Groups**

Entries in the Groups Menu are optional when creating a report. However, if your database files are indexed, grouping records provides an additional level of detail in the report.

The **Group on expression** option lets you organize your report by printing all records with the same value in a field or key expression under the same group heading. This is particularly useful if you want to create subtotals.

dBASE III PLUS starts a new group whenever it encounters a new value. Therefore, make sure that you INDEX or SORT the file on the same field or expression that you group records in the report. For example, you can group the report on the field Travelcode in Travel.dbf. Then, if you index the file by Travelcode, you can create a report listing all people who signed up for the same trip. You specify the fields you want to print under each group in the **Contents** option in the Columns Menu.

The **Group heading** option specifies the text printed on the report with each group expression (see Figure 5-5). For example, if the heading is "Travel Code," the report will print **Travel Code** followed by the name of the trip before it lists the people who are going on the trip.

The **Summary report only** option determines whether all the records for each group are displayed, or only their totals. The default value, NO, displays all contents and totals.

The **Page eject after group** option determines whether each group is printed immediately after the previous one, or is printed on a separate sheet of paper. The default value, NO, keeps the groups together on a page.

The **Sub-group on expression** option lets you further categorize groups of records in a report. For example, if the same travel package is repeated on different dates, you could INDEX the file on the Travelcode and Departure fields. Then, you could group the report by Travelcode, and specify the Departure date as a sub-group. The report will list in a sub-group all the people who have the same departure date.

The **Sub-group heading** option specifies the text introduction to the sub-group, for example, the subgroup heading of "Departure Date" would be displayed prior to the sub-group expression.

## Columns

You make entries in the Columns Menu to specify the fields on the report. A report can have up to 24 columns.

To delete an existing column, press **Ctrl-U**. To insert a new column, press **Ctrl-N.**

The **Contents** option specifies what goes in each column. This can be a field name or an expression.

If the contents is a field name, press ↵ to access the option, then press **F10** to activate a list of field names from which to choose. To select a field, highlight it and press ↵. Press → or ← to return to the main menu.

If the contents is an expression, type it in the entry area.

The **Heading** option identifies the contents of each column. You can enter up to four lines with up to 65 characters on each line. To begin making an entry, you need to press ↵ to open an entry area in which to create the column heading. Pressing **Ctrl-End** closes the entry area returning to the menu.

The **Width** option determines the size of the column. It automatically defaults to the field, expression, or heading size, whichever is larger (up to a maximum default of 30). You can increase or decrease this size. If the column is narrower than the field/expression or heading you entered, both will wrap when they are displayed.

The **Decimal places** option is for numeric values. It defaults to the number of decimal places specified in the file structure. If you decrease the size, the number is rounded.

The **Total this column** option is for numeric values, and allows you to choose whether you want to total numeric columns. The default value is YES, which causes subtotals to be displayed for each group and subgroup, and a grand total to be printed at the end of the report.

**Locate**

When you open the Locate Menu, a list of all the fields or expressions you previously defined in the Columns Menu appears. When you choose a column from this menu, that column entry is displayed for you to edit.

**Exit**

The Exit Menu gives you two ways to leave the menus.

The **Save** option stores a newly created report format (.frm) file or stores the changes made to an existing report file. The **Abandon** option discards a newly created .frm file, or abandons the changes made to an existing report format file. You can also abandon changes by pressing **Esc**.

## Page Layout

As you create a report by entering information in the Columns Menu, a coded diagram of the report is displayed on the screen. The F1 key alternates this display with a navigation key and help menu.

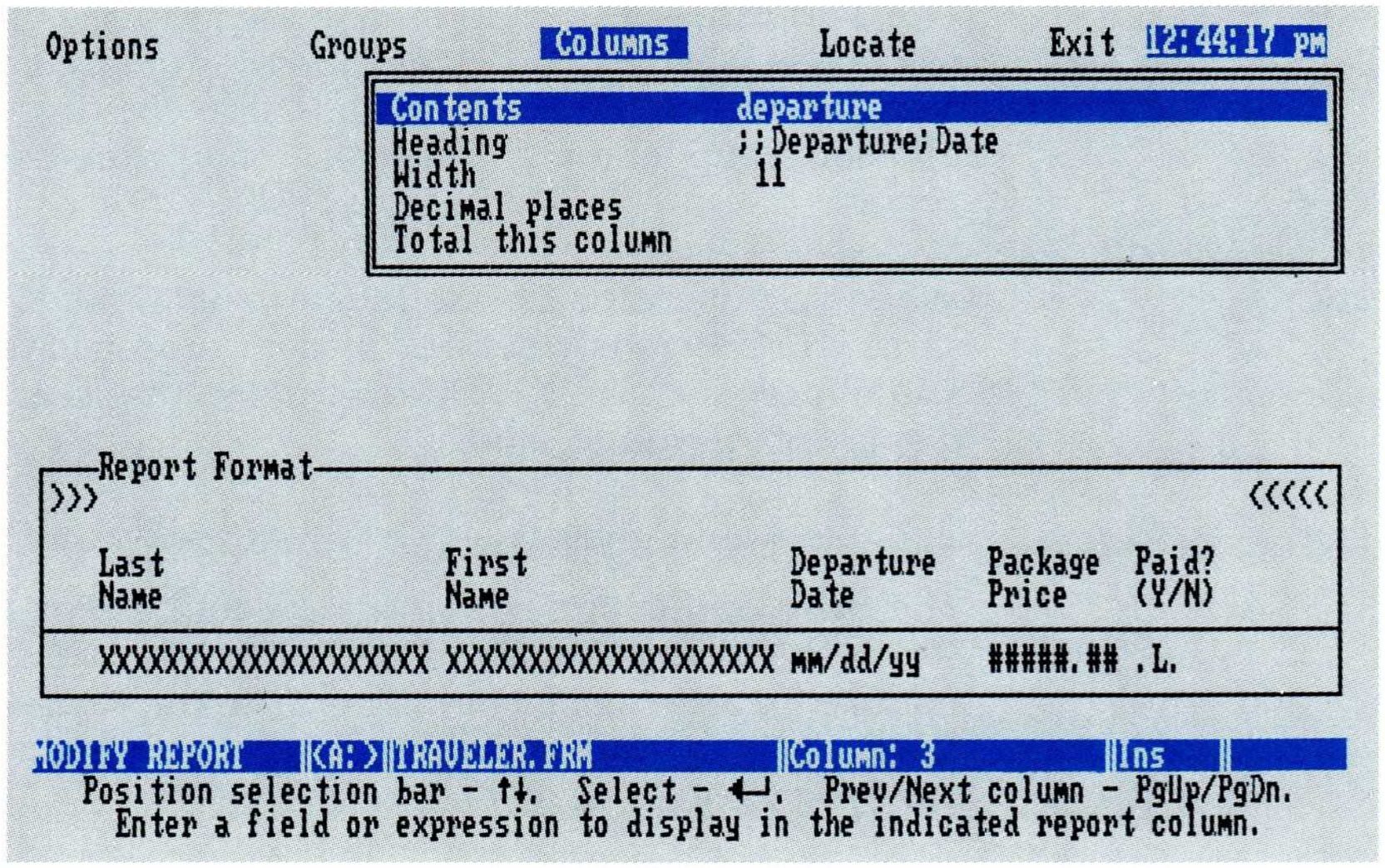

Options Groups Columns Locate Exit 12:44:17 pm
Contents departure
Heading ;;Departure;Date
Width 11
Decimal places
Total this column
Report Format
>>> <<<<<
Last Name
First Name
Departure Date
Package Price
Paid? (Y/N)
XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXX mm/dd/yy ######.## .L.
MODIFY REPORT (A:) TRAVELER.FRM Column: 3 Ins
Position selection bar - ↑↓. Select - ↵. Prev/Next column - PgUp/PgDn.
Enter a field or expression to display in the indicated report column.

Figure 5-6 On-screen report layout display

The codes are:

| Code | Meaning |
|---|---|
| < | Right margin |
| > | Left margin |
| X | Character column |
| 9 | Numeric column, not totaled |
| # | Numeric column, totaled |
| mm/dd/yy | Date column — if your date format is different, such as dd/mm/yy, that format displays. If SET CENTURY is ON, the year displays as yyyy. |
| .L. | Logical column |
| M | Memo column |
| ? | Indicates an undefined column |

Table 5-3 Report form page layout codes

# CREATE/MODIFY REPORT

The CREATE/MODIFY REPORT command has set ranges for entries in the report format file. These are:

| Option | Range |
|---|---|
| Page title | Four lines with up to 59 characters each. |
| Page width | 1 to 500 characters (the default is 80 characters). |
| Left margin | The default is 8; left margins are verified so that (page width − (left margin + right margin)) is greater than 0. |
| Right margin | The default is 0; the right margin is verified so that (page width − (left margin + right margin)) is greater than 0. |
| Lines per page | The default is 58; the range is 1 to 500. |

Table 5-4 Report format option ranges

## See Also

REPORT, SET FILTER

# CREATE/MODIFY SCREEN

## CREATE/ MODIFY SCREEN

CREATE/MODIFY SCREEN is a full-screen, menu-driven command you can use to create custom screen forms. It generates both a screen (.scr) and a format (.fmt) file. The CREATE/MODIFY SCREEN command provides "screen painter" functions allowing you to change the way fields appear on your screen when you use the APPEND and EDIT commands. While you are creating the custom screen form with this command, you can also modify the structure of the database file.

## Syntax

CREATE/MODIFY SCREEN < .scr filename > /?

## Defaults

Unless you specify otherwise, CREATE/MODIFY SCREEN creates or modifies a screen format file with the .scr file extension. When you save a screen format file, a format file with the .fmt file extension is created. If a catalog is open, and SET CATALOG is ON, the screen (.scr) file and format (.fmt) file are added to the catalog.

## Usage

You use the CREATE SCREEN command to create a new screen format file. You use MODIFY SCREEN < filename > /? to alter an existing screen format file.

If a catalog is open, you can type MODIFY SCREEN ? to display a list of screen (.scr) files associated with the active database file.

When you exit the Screen Menu with the **Save** option, dBASE III PLUS automatically generates a format (.fmt) file with the same name as the .scr file and activates the format file it generates. To deactivate an open format (.fmt) file, type SET FORMAT TO ↵ or open another format file.

# CREATE/MODIFY SCREEN

**WARNING**

- You specify a format (.fmt) file with the SET FORMAT TO command when you want to use the file with the full-screen APPEND or EDIT commands. However, you need to keep the screen format (.scr) file if you want to make changes to the entry screen's appearance. If you erase the screen format (.scr) file, you will not be able to use the CREATE or MODIFY SCREEN commands to change the .fmt file it created.
- You can use the MODIFY COMMAND command to make changes directly to a .fmt file, however, the changes will not be made to the .scr file.
- Do not use the double quote ('') character in your on-screen messages or text. The double quote character is reserved for use in format (.fmt) files to identify the start and end of character strings.

The CREATE/MODIFY SCREEN command allows you to:

- Create @...SAYs and @...GETs to position your text and fields.
- Create and modify a database (.dbf) file structure while you build a display or entry form.
- Change the width of fields on a form without affecting the .dbf structure.
- Define PICTURE clauses to include display and entry template symbols and functions.
- Define RANGEs for data entry.

- Draw lines and boxes on the display and entry form.
- Automatically generate a format (.fmt) file.
- Generate an ASCII text file for documenting the .scr file.

## The Blackboard

The area on which you create your custom screen form is called the blackboard. The blackboard extends from line 1 through line 20 on the screen. To move between the blackboard and the CREATE/MODIFY SCREEN menus, press **F10**. You use the Set Up and Modify Menu options, described below, to place text and fields on the blackboard.

### Editing Text

You use full-screen editing keys for changing text on the blackboard and to navigate around the blackboard screen.

### Repositioning Fields

To reposition a field on the blackboard, position the cursor on a field (highlighted area) and press ↵. Then move the cursor to a new position and press ↵ again.

### Changing the Size of a Field

To change the size of a field on the blackboard without changing its size in the file structure, move the cursor to the field.

- To enlarge the field size for the screen form, press **Ins**.
- To decrease the field size for the screen form, press **Del**.
- To alter the field size in the database file structure, press **F10** to display the **Width** option of the Modify Menu.

### Deleting a Field

To delete a field from the blackboard, move the cursor to the field and press **Ctrl-U**. dBASE III PLUS asks whether you also want to remove the field from the database file structure. This selection lets you modify the structure of a database (.dbf) file from the blackboard. Pressing **Ctrl-Y** deletes an entire line on the blackboard including the fields that are on that line.

## Options

The CREATE/MODIFY SCREEN menu bar offers four choices: **Set Up**, **Modify**, **Options**, and **Exit**.

### Set Up

This is the first menu that appears when you use the CREATE/MODIFY SCREEN command. It allows you to create a new database (.dbf) file or choose an existing database file from which to select fields to display on the blackboard. If a database file is already in use when you issue the CREATE/MODIFY SCREEN command, you don't have to select it again.

Choosing the **Select Database File** option activates a list of database files in the current catalog. If no catalog is active, the files on the default drive (or directory) are listed.

The Load Fields option puts all or selected fields from the active file on the blackboard. When you load fields, they are entered on the blackboard, one per line, starting on the first line. You can load additional fields onto the blackboard later, by positioning the cursor on the blackboard and pressing **F10**. Fields you select using the **Load Fields** option will be placed on the blackboard at the current cursor position.

To create a database file from the Set Up Menu, first specify a database filename. Then from the Modify Menu, you create fields in the new file, specifying the field name, width, and data type of each new field.

### Modify

The Modify Menu lets you enter field names on the blackboard and create or modify the structure of a new or existing database file. To move between the blackboard and the menu, press **F10**.

The **Action** option tells the Screen Painter whether to display text (@...SAYs) or edit a field (@...GETs). These are labeled as **Display/SAY** and **Edit/GET**. The default is **Edit/GET**.

The **Source** option lists the name of the active database file. This entry can be changed only through the Set Up Menu.

The **Contents** option displays a field name if the cursor is positioned on a field on the blackboard. Otherwise, the field name entry is blank. To choose an existing field name, press ↵ to display a list of available field names. When you choose a field, its Type, Width, and Decimals values are displayed.

If the cursor is not positioned on a field when you press **F10** to display the Modify Menu, the **Contents** option is blank. If you want to add a new field to the database file, select **<NewField>** from the field list menu. You will be prompted to specify the new field name, and you will have to specify the width and data type.

If you change the attributes of fields from the Modify Menu, you will also modify the field definition in the structure of the database file. You need to observe the same rules for modifying a database file structure as with the MODIFY STRUCTURE command.

The **Type** option defines the field contents as Character, Date, Logical, Memo, or Numeric. The Memo option is only active when **Edit/Get** is displayed.

The **Width** option defines the field size. You can change the field width for character and numeric fields from the value initially displayed. You cannot change the field width for date and logical fields because they have automatic default widths of eight and one, respectively.

The **Decimals** option is only active with numeric fields and defines the number of decimal places. The period counts as part of the overall field size.

The **Picture Function** and **Picture Template** options let you specify the type of input allowed in any field, and the way the field is displayed when the .fmt file is used. For example, you can limit the input to only numbers or letters, or you can specify that lower case input be converted to upper case. Choosing one of these options displays a menu list of template or function symbols you can use. The templates or symbols vary according to the data type of the field selected. For more information on using the PICTURE template and function symbols, refer to the description in this chapter of the @ command.

The **Range** option is active with numeric and date variables only. It specifies the lowest and highest numbers, or the range of dates, that dBASE III PLUS will accept when you are entering data using the format file.

## Options

The **Create text file image** option helps you document the contents of the screen (.scr) file. It converts the blackboard form to ASCII text. The ASCII text file includes information about each field, its location on the screen, and its PICTURE definition.

The **Single bar** and **Double bar** options let you draw a line or a box around any area of the screen form. When you choose one of these options, the blackboard appears and the Screen Painter prompts you to start and end a line or a box.

You can change the size of a box or line with the following procedure:

1. Place the cursor on any portion of the box or line you want to move.
2. Press ↵.
3. Move the cursor to a new position for the corner.
4. Press ↵ to redraw the box or line.

## Exit

The **Save** option saves the new .scr file or the changes made to an existing one. This automatically creates a .fmt file and activates the format file for use with full-screen APPEND and EDIT commands.

The **Abandon** option discards a new .scr file, or the changes made to an existing one. It does not generate a .fmt file. Selecting this option is equivalent to pressing **Esc**.

## Programming Notes

You can use CREATE/MODIFY SCREEN to set up a series of @...SAY...GETs for use in a dBASE program. After saving the file, you can copy the .fmt file into your dBASE program file, for example, using MODIFY COMMAND, and then add additional programming code.

## See Also

@...SAY...GET, CREATE <newfile>/MODIFY STRUCTURE, SET CATALOG, SET FORMAT

# CREATE < newfile > /MODIFY STRUCTURE

## CREATE < newfile > / MODIFY STRUCTURE

CREATE defines the structure of a new database file, stores the file in the specified disk directory and allows you to add new records.

MODIFY STRUCTURE modifies the structure of the active database file. Records are appended from the existing file into the modified structure. You are not prompted to add new records after saving the modified structure.

## Syntax

CREATE < newfile >

and

MODIFY STRUCTURE

## Defaults

CREATE creates a database file in the specified drive and directory location. If a drive and directory are not specified with the filename, the file is created in the current directory (or drive specified by SET DEFAULT TO). The database file is given the .dbf extension unless you specify a different extension.

You do not specify a filename with MODIFY STRUCTURE. This command can only modify the structure of the active database file.

Do not use the letters A through J or M as database filenames, because they are reserved for ALIAS names. However, you can, for example, specify AA as a database filename.

## Usage

CREATE < newfile > and MODIFY STRUCTURE provide the same display for creating or changing the database file structure. The structure defines each field in the database file (see Figure 5-7).

# CREATE < newfile > /MODIFY STRUCTURE

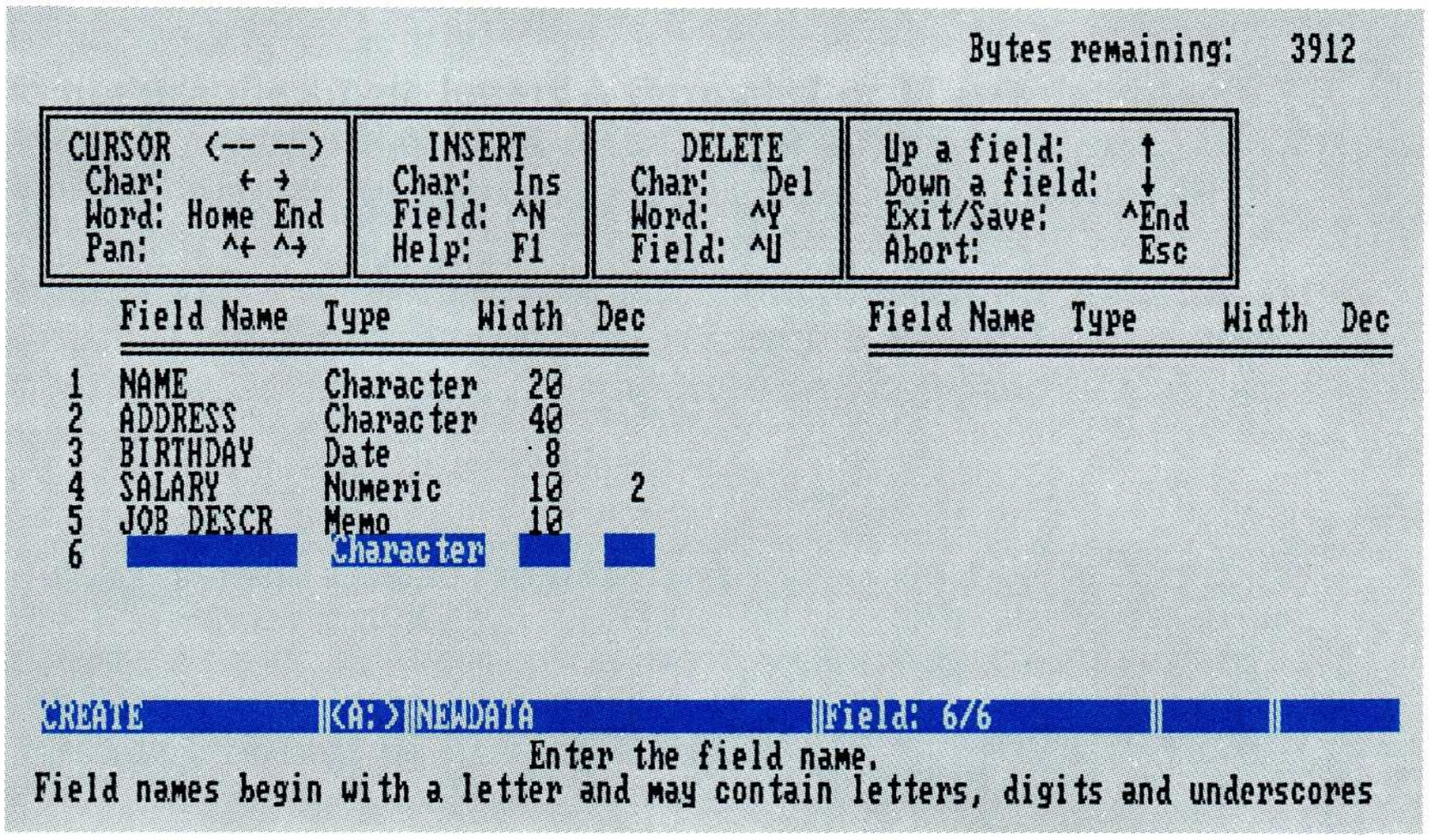



Figure 5-7 Creating a database file structure

## CREATE

You define the structure of a database file by specifying the following information for each field in the record:

Field Name
Type
Width
Decimal Places (for a numeric field)

The field name may be up to 10 characters in length, and may consist of letters, numbers, and the underscore character. The field name cannot contain embedded blank characters and the first character of a field name must be a letter. When you have finished entering the field name, press ↵.

You enter the field type by entering the first letter of the data type (**C**haracter,**N**umeric,**L**ogical, **D**ate, or **M**emo), or by pressing the **Spacebar** until the desired data type appears and then selecting it by pressing ↵.

A field width must be specified for numeric and character fields. This is the maximum number of digits or characters you intend to enter in the field. Character fields may be up to 254 characters long; numeric fields may be up to 19 digits including the sign and a decimal point.

Date, Logical, and Memo fields all have predefined widths. Date fields are always 8 characters and logical fields are always 1 character. Memo fields are automatically assigned a length of 10 in the database file, although the entry in a memo field may be up to 5,000 characters, or larger, depending on the word processor used. Data entered in memo fields is not stored in the database (.dbf) file but in an associated memo (.dbt) text file.

You can define a record having up to 128 fields. The maximum size of a record is 4,000 bytes. Each character position in a field takes up one byte. Memo fields use 10 bytes, even though the memo data is not actually stored in the database file. The 10 bytes identify the location of memo field entries in the memo file.

Instructions and error messages appear at the bottom of the screen as you create the database file. In addition, a cursor navigation menu appears at the top of the screen indicating the operations you can perform and the keys you can use. You can make the menu appear or disappear from the screen by pressing **F1**.

Below the menu, the entries for each field appear. Each field appears on a separate line containing the field name, the data type, the width, and the number of decimal digits (for a numeric field). A highlight appears on the current field, and a blinking cursor appears on the current entry column.

The arrow keys move the cursor to any field. When the fields take up more than a single screen, you can use the **PgUp** and **PgDn** keys to scroll forward or backward through the fields. When you create a new field, the cursor cannot be advanced beyond an incomplete field (or a field definition that is invalid), but can be moved backward to previous fields.

If you forget to include a field, and want to insert it, position the cursor where you want to insert the field and press **Ctrl**-N. The order in which you enter data in fields is determined by the field position in the database file, using the default data entry form dBASE III PLUS provides, or by the position of fields on a custom entry form created with CREATE/MODIFY SCREEN.

To delete a field, position the cursor on the appropriate field and press **Ctrl**-U. If at any point you want to abandon the file you are creating, press **Esc**.

When you're finished creating the database file structure, press **Ctrl-End** or press ↵ at the beginning of a new field. You will see a prompt asking you to confirm that you want to save the file. Press ↵ to save the file, or press any other key to resume. If you do save the file, you will next be asked whether you want to begin inputting records.

### MODIFY STRUCTURE

This command is used to modify the structure of an existing database file. You can use this command to add new fields, delete fields, or change a field's name, data type, or width. When you're finished making changes, you save the new structure by pressing **Ctrl-End**.

You can move through the displayed field name, width, type, and decimal columns using the ← and → keys, or you can press **Ctrl-**← and **Ctrl-**→ to move the highlight from field to field.

You can insert new fields using the MODIFY STRUCTURE command. Move the highlight to the field in the structure where you want to insert a field and press **Ctrl**-N. The field will be inserted in front of the highlighted field in the database file structure. Remember that the order in which you enter data in fields is determined by the position of fields in the database file using the default data entry form dBASE III PLUS provides, or by the position of fields on a custom entry form created with CREATE/MODIFY SCREEN.

MODIFY STRUCTURE provides an additional menu which you can display by pressing **Ctrl-Home**. Options in this menu help you quickly locate and go to specific fields when you have a large number of fields, or allow you to save or abandon changes to the structure. You select options from this menu by highlighting a selection on the menu bar and pressing ↵. The options provided are **Bottom**, **Top**, **Field #**, **Save**, or **Abandon**. The menu bar leaves the screen when you make a selection, or if you press **Ctrl-Home** again.

To exit MODIFY STRUCTURE, press **Ctrl-End** or the ↵ key when prompted to define a new field. If you press **Esc**, you can exit without saving any changes.

MODIFY STRUCTURE always makes a backup copy of the database file with a .bak extension. After the structure modifications are completed, dBASE III PLUS appends the contents from the backup file to the modified database file. dBASE III PLUS also creates a backup of the .dbt file with a .tbk extension. MODIFY STRUCTURE will not be able to create backup files if the disk or current directory is full. You should make sure that you have enough space available for the backup files before using MODIFY STRUCTURE.

**WARNING**
You should not change a field's name and its width or type at the same time. If you do, dBASE III PLUS will not be able to append data from the old field and your new field will be blank. If you want, change the name of a field and then save the file. Then use MODIFY STRUCTURE again to change the field's width or data type.

You also should not insert or delete fields from a database file and change field names at the same time. dBASE III PLUS appends data from the old file by using the position of fields in the file, if you change only field names. If you insert or delete fields, you could lose data because the order of fields is changed. You can, however, change field widths or data types at the same time as you insert or delete fields. In those cases, dBASE III PLUS appends data by field name, so the data will be appended correctly.

dBASE III PLUS will successfully convert data for a number of field type conversions. However, if you are changing field types, keep a backup copy of your original file, and check your new files to make sure the data has converted correctly.

If you convert numeric fields to character fields, numbers from the numeric fields will be converted to character strings. If you convert a character field to a numeric field, numeric characters in each record will be converted to digits until dBASE III PLUS encounters the first non-numeric character in each field. If the first character in a character field is a letter, the converted numeric field will be blank.

You can also convert logical fields to character fields, or vice versa. You cannot convert logical fields to numeric fields. You can also convert character strings which are formatted as a date (for example, mm/dd/yy or mm-dd-yy) to a date field, or convert date fields to character fields.

If you modify fields in a database file which are used in an index file key expression, you'll have to re-create the index file. If an index file is open when you modify a database file, it will automatically be closed when you save your changes to the database file.

# CREATE/MODIFY VIEW

## CREATE/ MODIFY VIEW

CREATE/MODIFY VIEW is a full-screen, menu-driven command that creates a file to link one or more related database files and their ancillary files.

## Syntax

CREATE/MODIFY VIEW < .vue filename > /?

## Defaults

Unless otherwise specified, dBASE III PLUS creates a view file with the .vue file extension. If a catalog is open, and SET CATALOG is ON, the .vue file is added to the catalog.

## Usage

You use the CREATE VIEW command to create a new view file. You use the MODIFY VIEW command to alter an existing view file. If you want to alter an existing view file from the current catalog, but you do not remember its name, type MODIFY VIEW ? This displays a list of view files in the open catalog. If you want to create a view file that is based on the current database files in use and a SET RELATION, use the CREATE VIEW FROM ENVIRONMENT command.

CREATE/MODIFY VIEW creates or modifies a view (.vue) file which defines the database files, index files, and the fields in a view, and sets up a relation between files (see SET RELATION). It also allows you to select a format file for display and entry of records, and to specify a filter condition to select records from the related database files you'll be able to see. When you want to use the relation and file setup in the view file again, issue the SET VIEW TO command. This command sets up the database and related files defined in the view file.

When you exit CREATE/MODIFY VIEW with the **Save** option, dBASE III PLUS automatically activates the new .vue file. To deactivate a .vue file, type CLOSE DATABASES. This closes all open .dbf, .ndx, and .fmt files. However, it does not close any open .cat file.

All .vue files contain the following elements:

- Database and associated index files and the work area selections for each of these files
- Relations between the files (if more than one database file is selected)
- Currently selected work area

Additionally, they can contain the following optional elements:

- Selected fields from each database file; with the CREATE/MODIFY VIEW command, these are set from the Set Fields Menu. With CREATE VIEW FROM ENVIRONMENT, these are set with the SET FIELDS TO command.
- A format (.fmt) file
- A filter condition

## Options

The CREATE/MODIFY VIEW menu bar offers five option choices: **Set Up**, **Relate**, **Set Fields**, **Options**, and **Exit**.

### Set Up

The Set Up Menu provides a list of all database (.dbf) files in the open catalog. If no catalog is open, it lists all database files on the default drive.

If a catalog is open, you can choose up to nine .dbf files to include in the .vue file. If it is not open, you can choose ten .dbf files.

To select a .dbf file, highlight the file and press ↵. If the file has associated index (.ndx) files, select the files you want to open from the list of index files that are displayed. Unless you are relating files by record number, you need to select an index file which contains fields in the relation expression entered in the Relate Menu.

dBASE III PLUS keeps track of any .dbf or .ndx files that are open when you issue the CREATE/MODIFY VIEW command. These files are marked for selection when you open the Set Up Menu.

### Relate

When you open the Relate Menu, all of the filenames selected in the Set Up Menu appear.

From the Relate Menu, you define the relations between the files selected in the Set Up Menu. As you select files, a chain effect occurs, with the first file selected as the initial link (also referred to as the *parent* file). Whenever you select a file, you must also specify the next file in the chain (referred to as the *child*), and the relationship between the two files. If another file is to be related, it is added to establish another parent-child relationship, and so forth.

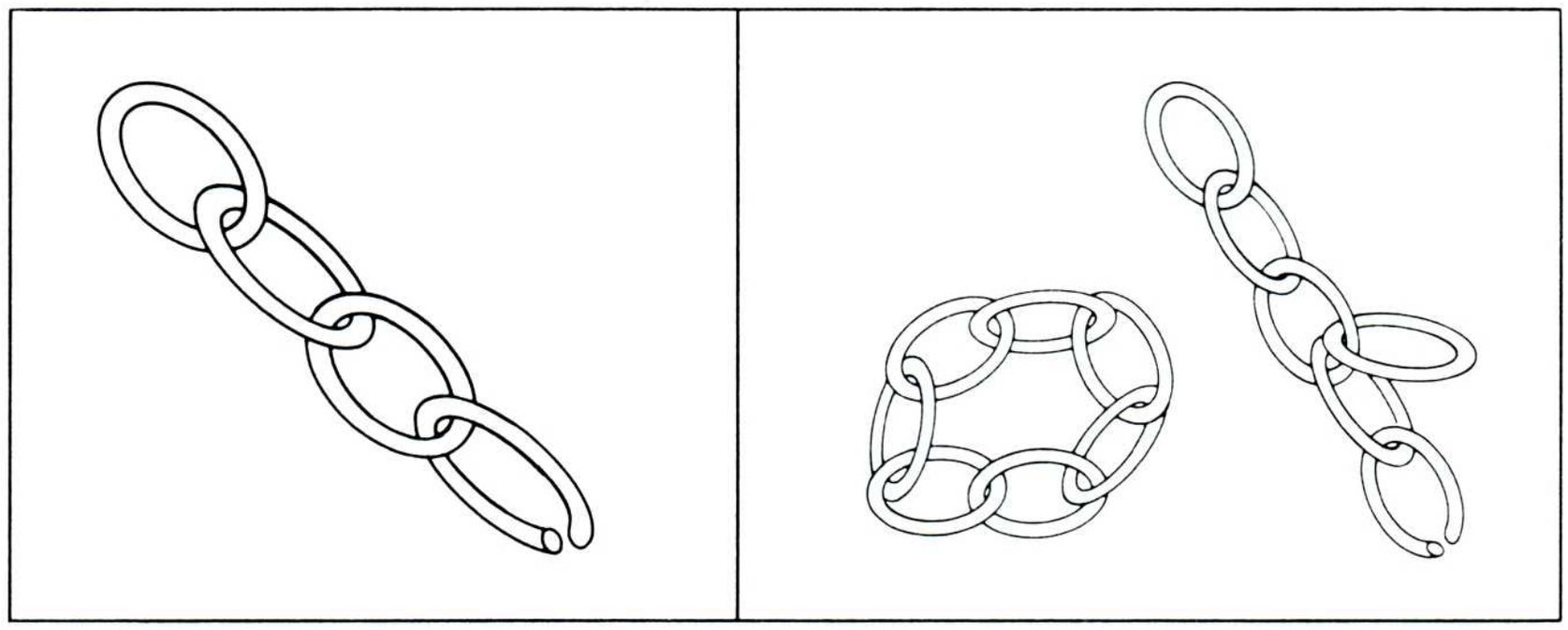


Figure 5-8 Proper and improper relations

As you create the relation between files, dBASE III PLUS displays the entire relation chain on the action line.

Follow these steps to create the relations, or chain:

1. Select the .dbf file that is to be the initial or controlling file, or the parent. This displays a list of all the other files selected in the Set Up Menu.

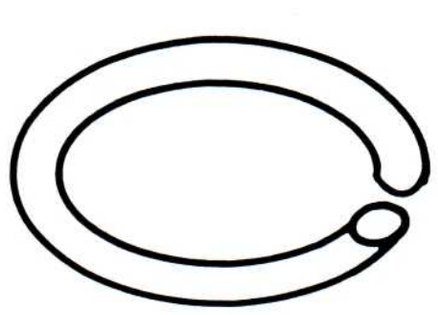


Figure 5-9 The parent file

2. Select the .dbf file that is to be the child. A highlighted entry area appears for you to enter the field or expression by which the new file is to be related to the preceding file.

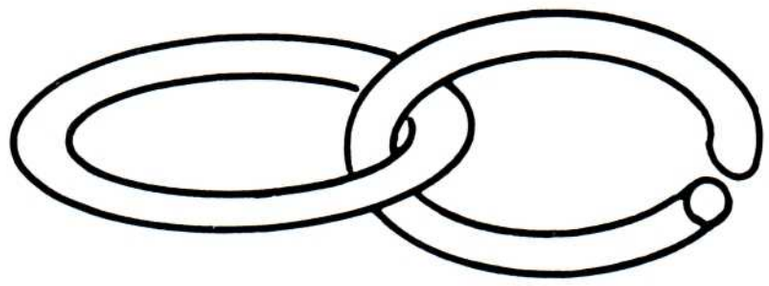


Figure 5-10 Two links

3. If you want to relate the files on a field, or want help selecting fields to include in the relating expression, press **F10** to display a list of possible field names from which to select. If the child is an indexed file, its key index expression is displayed on the message line to assist you. The relation expression must be of the same data type as the key index expression. However, if the child is not indexed, the expression must be numeric in order to relate the two files by record number. The most common numeric expression used to relate files is RECNO( ).
4. Press → or ← to return to the Relate Menu.

5. Either press → or ← to leave the Relate Menu, or follow steps 2 through 4 to create a new parent-child relation. If you are creating a new relation, you must select the last child in the relation chain to become the parent to a new child file. Otherwise, the relation chain is broken and the view file you're creating will be invalid. When you select the next database file as the parent file, a list appears containing all files that are not yet part of the relation chain.

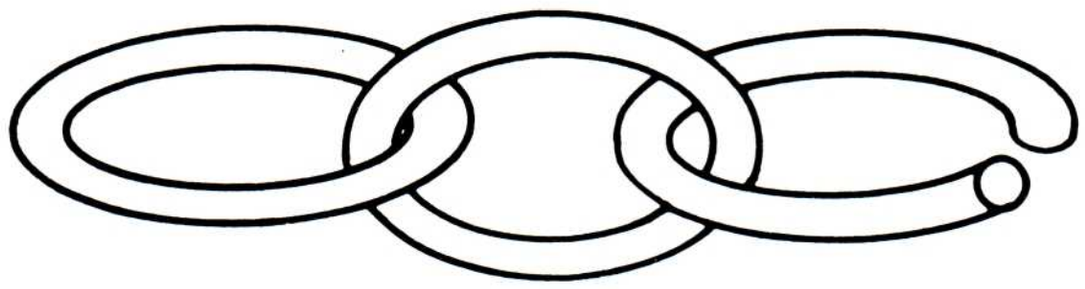


Figure 5-11 Extending the relation chain

6. Follow steps 2 through 5 until you complete your relation chain.

The entries in the CREATE/MODIFY VIEW menus are the same as issuing the following steps from the dot prompt:

```
SELECT 1
USE <file1>
SELECT 2
USE <file2> INDEX <ndxfile2>
SELECT 1
SET RELATION TO <file1> <Field name/expression> INTO <file2>
```

**Set Fields**

Opening the Set Fields Menu displays a list of all files in the relation chain from which you can select fields that will be accessible when the view file is in effect.

Choosing a filename displays a file structure list that includes all the field names in the database file. All fields are initially marked to be included in the view file. To unmark a field, move the highlight to the field name and press ↵. Close the menu list by pressing → or ←.

If you want to add an unmarked field to an existing view file, highlight the field and press ↵. The effect of marking fields is the same as building a fields list with the SET FIELDS TO command.

**Options**

The **Filter** option lets you specify a condition which must be met for any particular record to be included when the view file is used. See SET FILTER TO for details. The entry area scrolls horizontally, letting you enter an expression of up to 254 characters.

An important function of the **Filter** option is that it lets you set up a filter condition on fields other than those linking the database files. For example, two files, Travel and Clients, are linked by a common field, Lastname. You could set up a filter which includes only the fields for a Travelcode of CI10, by specifying the condition Travelcode = 'CI10'. If you want to specify a condition containing a field which is not in the current work area, you can specify the *alias* of the work area with the field, for example:

```
clients -> state = 'CA'
```

Refer to the SELECT command for a description of work areas and ALIAS names.

Choosing the **Format** option displays a list of format (.fmt) files in the open catalog, or on the default drive if no catalog is open. The format file you choose will always be in effect when you open the view file with the SET VIEW TO command.

If you want to create a .fmt file for a .vue file, save the .vue file and use CREATE SCREEN or MODIFY COMMAND to design the custom screen format. The format file should contain only fields that are accessible in the view. You can then return to the **Format** option of MODIFY VIEW and choose the new .fmt file from the menu list.

**Exit**

The **Save** option stores the newly created .vue file or saves changes made to an existing one. This automatically activates the .vue file.

The **Abandon** option discards a newly created .vue file, or disregards changes made to an existing one. This is equivalent to pressing **Esc**.

**Tips**

With the Files value set to 20 in Config.sys, you can have 15 files open. When you open a .vue file, make sure that you have enough room to open the files stored in the view: .dbf files, any associated .ndx and .fmt files, as well as space for the .vue file. The view file is only open as long as it takes to set up the other files.

**See Also** CREATE VIEW FROM ENVIRONMENT SELECT, SET CATALOG, SET FIELDS, SET FILTER, SET FORMAT, SET INDEX, SET RELATION, SET VIEW, USE

# CREATE VIEW FROM ENVIRONMENT

## CREATE VIEW FROM ENVIRON-MENT

CREATE VIEW FROM ENVIRONMENT builds a view (.vue) file which stores the setup of related files. The view file stores the selection of work areas; open database, format, and index files; set relations; set fields; and filter conditions.

## Syntax

CREATE VIEW < .vue filename > FROM ENVIRONMENT

## Defaults

Unless otherwise specified, dBASE III PLUS supplies the .vue extension. If a catalog is open, and SET CATALOG is ON, the view file is added to the catalog.

## Usage

CREATE VIEW FROM ENVIRONMENT builds a view (.vue) file with the following contents:

- All open database files, index files, and the work area of each file
- All relations between the database files
- The currently selected work area number
- The active field list
- The open format (.fmt) file, if any
- Filter conditions in effect

To deactivate the view file, open a different view file or type CLOSE DATABASES. This closes the open database files and their associated files that form the view file.

## See Also

SELECT, SET FIELDS, SET FORMAT, SET INDEX, SET RELATION, SET VIEW, USE

## DELETE

DELETE marks records in the active database file for deletion.

### Syntax

DELETE [ < scope > ] [WHILE < condition > ]
[FOR < condition > ]

### Defaults

Unless otherwise specified by the scope or the WHILE or FOR clause, only the current record is marked for deletion.

### Usage

With DISPLAY and LIST, records marked for deletion are indicated by an asterisk (*) in the first position of the record. You can unmark records already marked for deletion with RECALL.

With full-screen commands such as BROWSE or EDIT, records marked for deletion are indicated by **Del** on the status bar. In this mode, **Ctrl-U** both deletes and reinstates records.

### Record Pointer

DELETE with no parameters does not reposition the record pointer. Therefore, if you're at the end of the file, as after LIST or DISPLAY ALL, issuing DELETE has no effect.

### Examples

To mark only the first record in the Clients database for deletion:

```
. USE Clients
. DELETE
      1 record deleted
```

To mark record 6 for deletion:

```
. DELETE RECORD 6
      1 record deleted

. RECALL ALL
      2 records recalled
```

### See Also

DELETED(), PACK, RECALL, SET DELETED ON

## DIR

DIR displays the filenames, number of records, date of last update, and file size (in bytes) for each database file on the indicated disk drive. It is also possible to display the names of other types of files. DIR also shows the number of filenames displayed, the total number of bytes for the displayed files, and the total number of bytes remaining on the disk.

## Syntax

DIR [ < drive: > ] [ < path > ] [ < skeleton > ]

An alternate syntax is:

DISPLAY/LIST FILES [LIKE < skeleton > ] [TO PRINT]

## Defaults

Unless otherwise specified, only database file information for the default drive and directory is displayed.

## Usage

The directory will not reflect changes after records are added to or eliminated from a database file until the file is closed.

DISPLAY/LIST FILES are an alternative for DIR, except that they display files only in the current directory.

## Special Cases

If Config.sys is set to FILES = 20 and 15 files are open (including format files, program files, etc.), typing DIR displays **not readable** in the # records column.

## Examples

To display database file information:

```
. DIR
```

To display all the filenames on the disk:

```
. DIR *.*
```

To display index filenames:

```
. DIR *.NDX
```

To display filenames that are five characters long where D is the third character:

```
. DIR ??D??.*
```

If you have a directory named SALES on the default drive, you can display all disk filenames in that directory by typing:

```
. DIR SALES\*.*
```

To display only the database files in the SALES path:

```
. DIR SALES\
```

# DISPLAY

DISPLAY is used to view the contents of a database file.

## Syntax

DISPLAY [ < scope > ] [[FIELDS] < expression list > ]
[WHILE < condition > ] [FOR < condition > ] [OFF] [TO PRINT]

## Defaults

The current record is displayed, unless otherwise specified by the scope or WHILE or FOR clause.

All fields are displayed unless otherwise specified.

The record number is displayed unless the OFF option is included.

## Usage

If more than 20 records are displayed, DISPLAY pauses after every 20 lines. dBASE III PLUS then prompts **Press any key to continue...**

The contents of memo fields are not displayed unless explicitly named in the FIELDS expression list. Instead, the field name appears above the word Memo in the display. If memo fields are named in the expression list, their contents are displayed in 50-character wide columns. Use the SET MEMOWIDTH command to change this default display width.

At the end of the file (EOF( ) is .T.), DISPLAY shows headings only, as there are no remaining records to display.

When a record is DISPLAYed that is greater than 80 characters, the contents wrap around to the next line and may break up a field onto two or more lines.

## Special Cases

Each column has a heading. If the displayed item is the result of an expression involving a field (for example Cost/25), the column heading is the same as the expression.

The heading appears as you typed it. To receive all capital letters, type DISPLAY FIELD1. For upper and lower case headings, type DISPLAY Field1.

## Example

The following example uses the original version of the Tours database on the Sample Programs and Utilities disk. (If you have modified this file, your results may be slightly different.)

Display the travel code, the cost, and calculated commission. The first two items are fields in the database. The last item is an expression involving two database fields.

```
. USE Tours
. DISPLAY NEXT 5 Travelcode, Unitcost, Unitcost*Percent/100

Record#  Travelcode  Unitcost        Unitcost*Percent/100
      1  AV10          1378.00                   137.80000
      2  MA3            279.00                    20.92500
      3  JA6            579.00                    28.95000
      4  GE23          1969.00                   196.90000
      5  CN7            759.00                    56.92500
```

## See Also

SET HEADING, SET MEMOWIDTH

# DISPLAY HISTORY

## DISPLAY HISTORY

DISPLAY HISTORY outputs a list of commands that have been executed and are stored in a history buffer.

## Syntax

DISPLAY HISTORY [LAST < expN > ] [TO PRINT]

## Defaults

The default number of commands stored in the history buffer is 20. You can alter this number with SET HISTORY TO.

## Usage

DISPLAY HISTORY lets you view previously executed commands from the least recent to the most recent. The listing halts whenever a screen is filled.

## Options

DISPLAY HISTORY displays all commands that are in the history buffer unless you limit the number by including the LAST option. In this case, it displays only the last < n > commands.

## See Also

LIST HISTORY, SET DOHISTORY, SET HISTORY

## DISPLAY MEMORY

DISPLAY MEMORY shows the name, type, size, and status of each active memory variable. It also shows the number of active memory variables, the number of additional variables possible, the amount of memory used, and the amount of memory still available for memory variables.

### Syntax

DISPLAY MEMORY [TO PRINT]

### Usage

A maximum of 256 memory variables may be active at one time.

A maximum of 6,000 bytes of memory is used for memory variables, unless otherwise specified in a Config.db file.

If the number of active memory variables exceeds a screenful, DISPLAY MEMORY pauses after every screenful and displays the prompt **Press any key to continue...**

The display for numeric variables contains both the number as it appears in displays and the internal representation of the variable.

### Examples

To illustrate, the character string "Hi there" has been stored to the variable Greetings; the number 400 has been stored to Number; and a logical True (.T.) has been stored to the variable Truth.

```
. DISPLAY MEMORY
GREETINGS   pub    C   "Hi there"
NUMBER      pub    N         400 (      400.00000000)
TRUTH       pub    L   .T.
    3 variables defined,       21 bytes used
  253 variables available,   5979 bytes available
```

### See Also

PRIVATE, PUBLIC

# DISPLAY STATUS

## DISPLAY STATUS

DISPLAY STATUS provides information about the current dBASE III PLUS session and pauses at each full-screen display. LIST STATUS displays the same information but scrolls without pausing.

## Syntax

DISPLAY STATUS [TO PRINT]

## Usage

For each open database file, DISPLAY STATUS shows the following information:

- Database name with drive and directory location
- Work area number and Alias
- Database relations
- Open index filenames and index key expressions
- Open memo filenames
- Filter formulas
- Format files

It also shows:

- Loaded modules
- Currently selected work area
- ON command settings
- Current file search path
- Default disk drive
- Print destination
- Currently open PROCEDURE file
- Current settings for most ON/OFF SET commands
- The setting for DEVICE (SCREEN or PRINT)
- Left margin setting
- Function key assignments

## See Also

LIST STATUS

## DISPLAY STRUCTURE

DISPLAY STRUCTURE provides the following information about the active database: the filename, its drive and directory location, the number of data records, the last date that changes were made, the definition of each field, and the number of bytes in a record.

### Syntax

DISPLAY STRUCTURE [TO PRINT]

### Usage

The total number of bytes in a record is the sum of all the field widths plus one (the extra byte stores the deleted record marker).

If the number of fields in the database exceeds 16, DISPLAY STRUCTURE pauses after every 16 field names and displays the prompt **Press any key to continue...**

DISPLAY STRUCTURE shows a > symbol beside each field in a SET FIELDS list if SET FIELD is ON.

### Examples

The following example uses the Clients database on the Sample Programs and Utilities disk.

```
. USE Clients
. DISPLAY STRUCTURE
Structure for database : B:Clients.dbf
Number of data records :      49
Date of last update    :  07/24/85
Field  Field name   Type       Width   Dec
    1  FIRSTNAME    Character     20
    2  LASTNAME     Character     20
    3  ADDRESS      Character     25
    4  CITY         Character     20
    5  STATE        Character      2
    6  ZIPCODE      Character      5
    7  PHONE        Character     13
** Total **                     106
```

### See Also

LIST STRUCTURE

## DO

DO causes a dBASE III PLUS command file or procedure to be executed. It may also pass parameters to the named program.

## Syntax

DO <filename> [WITH <parameter list>]

## Defaults

The filename must include the drive designator if the file is not on the default drive, unless a path is set. A .prg extension is assumed unless otherwise specified.

## Usage

Each DO <filename> counts as an open file. The total number of open files (database files, index files, format files, command files, and so on) cannot be more than 15 (FILES = 20). If you are using a procedure file, only the one named with SET PROCEDURE TO <filename> counts as an open file.

When the program called by DO is complete, control returns to the calling program, or to the keyboard if the DO was issued from the dot prompt.

Memory variables created in called programs must be declared PUBLIC if they're used after the called program terminates. All PRIVATE memory variables created within programs are released once the program terminates.

## Options

The WITH option allows passing of parameters to the subroutine. The parameter list can contain any valid dBASE III PLUS expressions. To specify a field name to take precedence over a memory variable with the same name, specify the alias name of the work area in which the database file is open.

## Examples

Assume that you have a program named Taxes.prg which is located on drive B, while the default drive is A. To execute the program, type:

```
DO B:Taxes
```

The following program, Areacalc.prg, calculates the area of a rectangle based on the formula Area = Length * Width:

```
* The command file Areacalc.prg
PARAMETERS Length,Width,A
* A corresponds to area
A = Length * Width
RETURN
* EOF: Areacalc.prg
```

To pass the parameters, width and length, as 4 and 6 respectively, and return the correct value in the variable area type:

```
. Area = 0
0
. DO Areacalc WITH 6,4, Area
                    24
. ? area
      24
```

## See Also

MODIFY COMMAND, PARAMETERS, PRIVATE, PROCEDURE, PUBLIC, SET PROCEDURE

# DO CASE

DO CASE is a structured programming command that selects only one course of action from a set of alternatives.

## Syntax

```
DO CASE
    CASE <condition>
        <commands>
    [CASE <condition>]
        <commands>
    [OTHERWISE]
        <commands>
ENDCASE
```

## Usage

ENDCASE terminates the DO CASE structure. Command pairs such as DO CASE...ENDCASE, IF...ENDIF, and DO WHILE...ENDDO must be properly nested within DO CASE. Nested DO CASEs are permitted.

CASE <condition> sets up a conditional statement for evaluation. It is a logical expression such as A = B or numvar <11. When <condition> evaluates to logical True, all subsequent commands are carried out until any one of the following commands is reached: another CASE, OTHERWISE, or ENDCASE.

After one True CASE is found and its associated commands processed, no further CASE statements are evaluated. dBASE III PLUS skips immediately to the first command after ENDCASE. If no CASE statements evaluate to True, and there is no OTHERWISE statement, the program processes the first command following ENDCASE. OTHERWISE causes the program to take an alternative path of action when *all* CASE statements evaluate to False.

## Tips

In instances where only one of many conditions will evaluate to True, or where only the first True instance is to be processed, the DO CASE command is preferable to the IF command.

The CASE construct is often used when there are a small number of exceptions to a condition. The CASE <condition> statements can represent the exceptions, and the OTHERWISE statement the more common situation.

## See Also

DO, DO WHILE, IF, MODIFY COMMAND

## DO WHILE

DO WHILE is a structured programming command that allows command statements between it and its associated ENDDO to be repeated as long as the specified condition is True.

### Syntax

```
DO WHILE <condition>
    <commands>
   [EXIT]
    <commands>
   [LOOP]
    <commands>
ENDDO
```

### Usage

ENDDO must terminate a DO WHILE structure.

EXIT transfers control from within a DO WHILE...ENDDO loop to the command immediately following the ENDDO.

LOOP returns control to the beginning of a DO WHILE...ENDDO program structure. LOOP prevents the execution of the remaining commands in the DO WHILE construct.

Any structured commands within a DO WHILE...ENDDO structure must be properly nested. Nested DO WHILEs are permitted.

Comments entered following ENDDO on the same line are ignored.

DO WHILE <condition> opens a structured procedure that processes subsequent commands only while the condition evaluates to True, for example, .NOT. EOF() .AND. mvar1 = 11.

If <condition> evaluates to a logical True, all subsequent commands are carried out until an ENDDO, LOOP, or EXIT is encountered. ENDDO and LOOP return control to the DO WHILE command for another evaluation of <condition>. EXIT passes control to the statement following the ENDDO.

If the condition evaluates to a logical False, dBASE III PLUS skips all commands between DO WHILE and ENDDO and goes to the command following ENDDO.

## Programming Notes

You can use macros in the conditional portion of a DO WHILE loop only if the value of the variable in the macro does not change, as the DO WHILE statement is parsed only the first time through the loop. After the first time, the DO WHILE statement is executed from memory. Also, macro substitution must be within the lowest nested level of a program.

## Examples

The first example shows how to correctly use a macro in the conditional portion of a DO WHILE loop executing in a dBASE program file:

```
* This is an example of the correct way
* to use a macro in a DO WHILE statement.
*

USE Clients INDEX Cnames
STORE [UPPER(Lastname) = "KEEGAN"] TO condition
FIND Keegan
DO WHILE &condition .AND. .NOT. EOF()
   * The value of condition never changes
   ? TRIM(Firstname), Lastname
   SKIP
ENDDO
USE
RETURN
```

The next program file example illustrates the correct use of the EXIT command. The program lets you view five records from the Tours database file and then decide whether you want to see the next five. Once you've found the information you need, you can tell dBASE III PLUS to EXIT the program.

```
* Start .prg file
USE Tours
DO WHILE .NOT. EOF()
   CLEAR
   LIST NEXT 5 Travelplan, Departure, Unitcost
   ?
   WAIT "Press X to stop, any other key to continue" TO mlist
   IF mlist $ "Xx"
      EXIT
   ENDIF
   SKIP
ENDDO
* EOF .prg
```

The next program example creates a menu designed so that if you enter an invalid choice, the LOOP command causes dBASE III PLUS to return to the DO WHILE and you can try again. Without LOOP, dBASE III PLUS would first go through the DO CASE commands and, after finding no matching CASE conditions, would return to the DO WHILE.

In the example, the substring search operator, $, is used in the command IF Mlist $ "Xx". The IF command determines whether the character stored in Mlist is either X or x.

```
* Beginning of .prg file
USE Clients
DO WHILE .T.
   mchoice = " "
   CLEAR
   @  5,5 SAY "A - APPEND Records"
   @  7,5 SAY "E - EDIT Records"
   @  9,5 SAY "R - Return to dot prompt"
   @ 11,5 SAY "Enter choice " GET mchoice PICTURE "!"
   READ
   IF .NOT. mchoice $ "AER"     && If contents of mchoice are not A, E, or R
      LOOP
   ENDIF
   DO CASE
      CASE mchoice $ "A"           && Append if mchoice is A
         APPEND
      CASE mchoice $ "E"           && Edit if mchoice is E
          EDIT
      CASE mchoice $ "R"           && Exit if mchoice is R
          EXIT
   ENDCASE
ENDDO
CLOSE DATABASES
* End of .prg file
```

## See Also

DO, DO CASE...ENDCASE, IF...ENDIF, RETURN

## EDIT

EDIT is a full-screen command used to alter the contents of a record in the active database file.

## Syntax

EDIT [ < scope > ] [FIELDS < list > ] [WHILE < condition > ] [FOR < condition > ]

## Defaults

If EDIT is used without parameters, the current record is EDITed.

## Usage

EDIT uses the standard full-screen cursor control keys. The arrow keys move the cursor within a record. **PgUp** backs up to the previous record. **PgDn** advances to the next record. To exit and save all changes, press **Ctrl-End**. Press **Esc** to save changes to all but the current record.

To EDIT a memo field, enter **Ctrl-PgDn** when the cursor is positioned at the beginning of the field. The control key functions are the same as for MODIFY COMMAND. To exit from this mode, enter **Ctrl-End** to save changes or **Esc** to exit and abandon changes to the current record.

EDIT and CHANGE are identical commands.

## Example

To EDIT the 21st record in the Clients database file:

```
. USE Clients
. EDIT RECORD 21
```

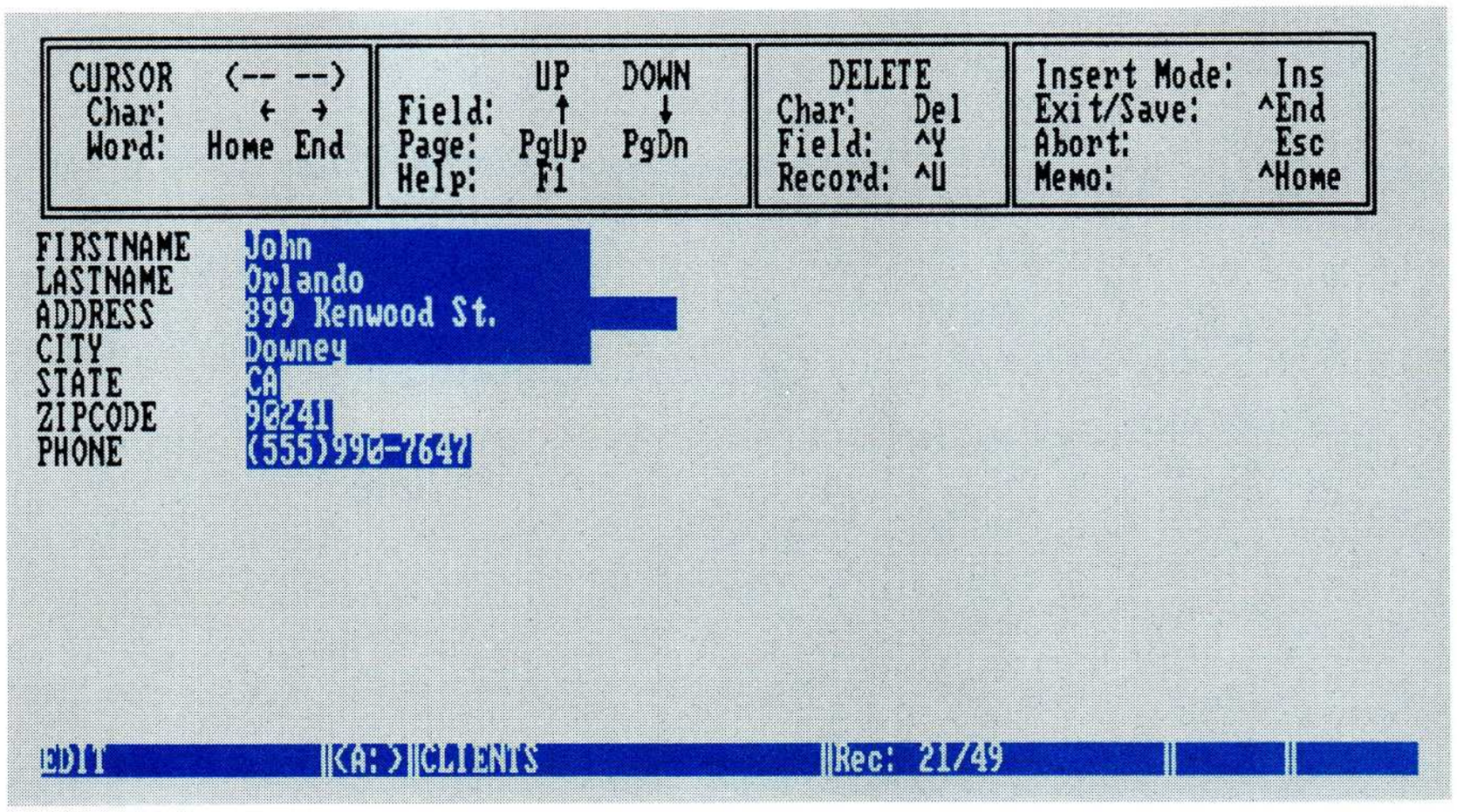



Figure 5-12 The EDIT command

## See Also

CHANGE, MODIFY COMMAND, SET FIELDS, SET FORMAT TO

## EJECT

EJECT causes the printer to advance the paper to the top of the next page.

## Syntax

EJECT

## Usage

EJECT issues a form feed (ASCII character value of 12) to the printer. For proper printer operation, the paper must initially be set to the top-of-form. Refer to your printer manual for instructions on how to set top-of-form.

Issuing an EJECT when the printer is either not connected or not turned on may "lock" the computer. Connecting and turning on the printer should "unlock" it. Otherwise, the computer must be restarted, which can result in data loss. Consult your computer operating manual for instructions.

EJECT resets PROW() and PCOL() to zero.

## See Also

PCOL(), PROW(), SET DEVICE, SET PRINT

## ERASE

ERASE removes a file from the disk directory.

### Syntax

ERASE <filename> /?

or

DELETE FILE <filename> /?

### Defaults

The filename must include the file extension. If the file is not on the default drive, include the drive designator.

### Usage

Use ERASE ? to display a menu of cataloged files.

You may not delete an open file.

If you ERASE a database file (.dbf) that has memo fields, you must delete the .dbt file that contains the memo fields separately.

Unlike ERASE in PC-DOS, dBASE III PLUS does not permit the use of wildcard characters.

DELETE FILE is the same as ERASE.

### See Also

CLOSE, USE

## EXPORT

EXPORT exports the open dBASE III PLUS database file to a file format useable by pfs:File. The records are exported in an indexed order if an index file is in use. A screen form will also be exported if a format file is in use.

### Syntax

EXPORT TO <filename> [TYPE] PFS

### Defaults

The file has the same name as the dBASE III PLUS database (.dbf) file from which it was made. pfs:File forms normally do not have a file extension, however, you can provide one.

### Usage

This command acts on an active database file *and* its ancillary files, merging them into a single pfs:File form. The original dBASE III PLUS files remain intact after an EXPORT.

The format of the fields in the newly created file depends on whether a format file is engaged when you EXPORT. If no format (.fmt) file is SET, EXPORT uses the dBASE III PLUS default entry format provided for commands such as APPEND and EDIT.

If the dBASE III PLUS database file was previously created with IMPORT, it has an associated format (.fmt) file. If a format file is selected using the SET FORMAT TO command, EXPORT uses that form. If the database file has several format (.fmt) files, EXPORT uses whichever form is open.

When you EXPORT a format file, check that it does not contain more than 200 @ commands. Also, the form should not specify more than 21 rows. The rows specified on your form must be between row 0 and row 20.

If a TO file already exists and SET SAFETY is ON, you are warned about overwriting. If SET SAFETY is OFF, the existing PFS file is overwritten automatically, even if you have an unacceptable format file.

**NOTE**
dBASE III PLUS allows you to build files with fields that may be larger than your other software can accept. Although these fields are exported, they may be cut short by the other program. Note the limitations of other programs before creating files for them with EXPORT.

## Examples

To EXPORT the Tours.dbf and Tours.fmt database files, you would type the following (please note that the creation of this file requires a minimum of 16,384K bytes free on your disk):

```
. USE Tours
. SET FORMAT TO Tours
. EXPORT TO Ptours TYPE PFS
```

## See Also

COPY TO, SET FORMAT, SET SAFETY

## FIND

FIND searches an indexed database file for the first record with an index key that matches the specified character string or number. FIND conducts a very rapid record search.

## Syntax

FIND <character string> / <n>

## Tips

SEEK has a similar function but, unlike FIND, it can search for an expression.

LOCATE also has a similar function but doesn't require an indexed file.

## Record Pointer

If the specified character string or number is not found, the message **No Find.** displays on the screen. SET TALK OFF suppresses this message. The record pointer moves to the end of the file, EOF( ) is True, and FOUND( ) will return False.

## Special Cases

FIND works without delimiters if the character string for which you are searching does not contain leading blanks and does not begin with a character normally used as a delimiter. If there are leading blanks, the specified string must include the exact number of leading blanks and be enclosed in single or double quotation marks, or square brackets. If a character string begins with a delimiter character, for example, *'two*, you need to specify FIND with delimiters, and choose a different delimiter than that in a character string.

When searching for the contents of a character memory variable, the variable must be used with the & (Macro) function. If the memory variable contains leading blanks, you must enclose the function in quotation marks: "& <memvar>".

## Examples

Use the file Clients, indexed on the field Lastname, for this example. To find the first record with a last name beginning with the upper-case letter G:

```
. USE Clients
. INDEX ON Lastname TO Clast
  100% indexed          49 records indexed
. FIND G
. ? LASTNAME
Garnett
. SET EXACT ON
. FIND G
No find.
. SET EXACT OFF
```

To search for a record for Zychowoski (there is no such record):

```
. FIND Zychowoski
No find.
. ? EOF()
.T.
```

## See Also

EOF(), FOUND(), INDEX, SEEK, SET INDEX, SET ORDER

## GO/GOTO

GO/GOTO positions the record pointer to a specified record in the active database file.

### Syntax

[GO/GOTO] < expN >

or

GO/GOTO TOP/BOTTOM

### Usage

If an index file is in use with the database, TOP and BOTTOM refer to the first and last records in the index file: GO < n > refers to the specified record number, and not to a position in the index file.

With SET DELETED ON, you may GOTO a record that is marked for deletion by directly specifying its record number.

### Examples

The Clients database from the Sample Programs and Utilities disk is used in the following examples.

To position the record pointer to record 4:

```
. USE Clients
. 4
. ? RECNO()
        4
```

To position the record pointer to record 5:

```
. STORE 3 TO mnum
3
. GO mnum + 2
. ? RECNO()
        5
```

To position the record pointer to the top of the database file:

```
. GO TOP
. ? RECNO()
        1
```

## HELP

HELP is a menu-driven command that provides information about the commands in and use of dBASE III PLUS.

## Syntax

HELP [ <keyword> ]

## Usage

The HELP command uses the Help.dbs file on the System #2 disk supplied by Ashton-Tate.

The keyword must be a dBASE III PLUS command, function, or HELP screen name. You can press **F1 Help** instead of typing HELP to get the Help Main Menu. This displays the main Help menu. Use arrow keys to move through the menu. Press ↵ to choose the highlighted option or enter the number of the option beside the word ENTER.

To exit Help, press the **Esc** key.

## Examples

To display Help, type:

```
. HELP
```

To activate Help for a specific command such as COPY, type:

```
. HELP COPY
```

## See Also

SET HELP

## IF

IF is a structured programming command that enables conditional processing of commands. It must terminate with ENDIF.

## Syntax

```
IF <condition>
    <commands>
[ELSE]
    <commands>
ENDIF
```

## Usage

Any structured commands within an IF...ENDIF structure must be properly nested. Nested IFs are permissible.

The space following ENDIF on a line may be used for comments.

IF <condition> sets up a conditional statement which is a logical expression such as (A = B or Numvar <11) for evaluation. If <condition> evaluates to logical True, all subsequent commands are carried out until an ELSE, or the ENDIF if no ELSE exists, is reached. dBASE III PLUS then executes the first command after ENDIF.

If <condition> evaluates to logical False, dBASE III PLUS goes directly to the ELSE or ENDIF, whichever it encounters first. ELSE causes all subsequent commands to be carried out until ENDIF is reached.

If there are multiple IFs in a command structure, ELSE refers to the IF immediately preceding it.

## See Also

DO CASE, IIF()

## IMPORT

IMPORT creates dBASE III PLUS files from pfs:File forms.

## Syntax

IMPORT FROM < filename > [TYPE] PFS

## Defaults

The filename must include the file extension if one exists, although files imported from pfs:File do not usually have an extension. The newly created dBASE III PLUS files are given the same name as the pfs:File form. If a catalog is open, the new file is added to the catalog. Records created in the new database file are limited to a maximum of 4,000 bytes.

IMPORT will locate a file on a drive or directory specified with a dBASE path (SET PATH TO command). If no drive or path is specified, dBASE III PLUS assumes that the file to be imported is on the default drive and directory.

## Usage

To make sure that you import data correctly, check your pfs:File form before using the IMPORT command for the following conditions:

- 128 data items per form. You can have up to 128 fields in a record. Notice, however, that headings and comments in your pfs:File form are also converted to fields.
- 254 characters per data item. 254 is the maximum length for character fields in dBASE III PLUS.

IMPORT automatically separates the IMPORTed file into several files understood by dBASE III PLUS: a database (.dbf), format (.fmt), and a view (.vue) file. All have the same filename, and are assigned default file extensions by dBASE III PLUS.

dBASE III PLUS creates the view file by internally performing the equivalent of the following commands at the end of its IMPORT procedure:

```
USE < filename >
SET FORMAT TO < .fmt filename >
CREATE VIEW < .vue filename > FROM ENVIRONMENT
```

By performing an internal CREATE VIEW FROM ENVIRONMENT command, dBASE III PLUS creates a view (.vue) file that contains both the .dbf file and the .fmt file. All files are entered in the catalog, if one is open.

In future sessions, open the file with either USE or SET VIEW TO. If you use SET VIEW, both the database file and the format file are opened. USE only opens the database file; to use the format file, you must also specify the SET FORMAT TO command.

When IMPORT creates a format file, it may create @ statements with column references outside the range of 0 to 79. If so, you need to use MODIFY COMMAND or another word processor to modify the file. If you use IMPORT to overwrite an existing database (.dbf) file, answer **Yes** to all three overwrite prompts that appear on your screen. This ensures that the created database, format, and view files can still be used together.

## Examples

Assume you have pfs:File data in a file named Agents. To create dBASE III PLUS files from it, you would type:

```
. IMPORT FROM Agents TYPE PFS
```

The IMPORT command creates the necessary files and puts them in use. To display information about the imported file use the DISPLAY STATUS and DISPLAY STRUCTURE commands.

## See Also

APPEND FROM, CREATE VIEW, EXPORT, SET FORMAT, SET VIEW TO, USE

## INDEX

The INDEX command creates a file in which index keys from an associated database file are ordered alphabetically, chronologically, or numerically. The index file contains the index key values and the corresponding record number for each record in the database file.

### Syntax

INDEX [ON <key expression> TO <filename> [UNIQUE]]

### Defaults

Unless otherwise specified as part of the filename, the default drive and an .ndx extension are assumed.

If you only type INDEX ↵, dBASE III PLUS prompts for the index expression and the index filename .

When SET SAFETY is ON (the default), dBASE III PLUS displays a warning prompt before overwriting an index file with the same name as the new index file.

### Usage

You specify an index key expression to order the records of an associated database file. INDEXing always occurs in ascending order. The database file records are not physically rearranged by the INDEX operation. Instead, the order is maintained by the index (.ndx) file, which must then be used with the database file for the records to appear in the desired order.

The index key expression may be a single field or any valid dBASE expression, however, it may not contain logical or memo fields. The maximum length of the index key expression is 220 characters. The maximum length of the index key, the length of the result of the evaluated index key expression, is 100 characters.

The data type of the index key expression determines whether records will be ordered chronologically (by date), alphabetically (character strings), or numerically.

When the index key expression includes several fields, they must all be converted in the expression to the same data type. Place the most important field first and then convert the remaining fields to create an expression that matches the data type of the first field. You can also use dBASE functions to convert your fields to the correct type. Some of the functions most commonly used in creating index key expressions are STR(), CTOD(), DTOC(), YEAR(), MONTH(), DAY() and VAL().

The index file contains the index key values (the key expression evaluated) and the corresponding record number for each record in the database file. The index file maintains the order of records in a database file as long as the index file is open. Up to seven index file database files can be open at one time, however, the first one in a list controls the order in which records will appear (unless you use the SET ORDER TO command).

## Options

The UNIQUE option is the same as issuing SET UNIQUE ON before an INDEX. When several records have the same value on the key field, only the first record that dBASE III PLUS encounters with that value is included in the .ndx file.

Whenever you REINDEX an .ndx file that was created with UNIQUE, the file retains its UNIQUE status, regardless of whether SET UNIQUE is ON or OFF.

## Tips

You can specify index expressions that convert field types. For example, if you wanted to INDEX a date field chronologically and have the last names in alphabetical order for each date, you could create the following expression:

INDEX ON STR(YEAR( < date > ),4) + STR(MONTH( < date > ),2) + STR(DAY( < date > ),2) + Last_name TO < index filename >

In this example, the date field is first converted to a numeric type using the YEAR(), MONTH(), and DAY() functions. Next, the numeric data is converted to character type data with the STR() function. Because Lastname is a character type field, the index key is also of character field type.

## Examples

The next example shows the date field, Reservdate, and the Lastname field in the Travel database file. The unindexed Travel.dbf file looks like this:

```
. USE Travel
. LIST Reservdate,Lastname
Record#  Reservdate Lastname
      1  07/15/85   Buckman
      2  07/15/85   Lisbonn
      3  07/15/85   Bicksby
      4  07/15/85   Garnett
      5  07/15/85   Kafmanan
         .
         .
         .
     49  11/23/85   Young
```

To index the file, listing all reservations alphabetically by the last name of the passenger, type:

```
. INDEX ON Lastname TO Passname
   100% indexed          49 records indexed
. LIST Reservdate, Lastname
Record#  Reservdate Lastname
     15  07/16/85   Adams
     29  07/18/85   Anderson
      9  07/15/85   Arlich
     42  07/19/85   Beman
      3  07/15/85   Bicksby
         .
         .
         .
      8  07/15/85   Zambini
```

To index the file, listing all passengers by their phone number exchange, and in alphabetical order by last name, type:

```
. INDEX ON Phone + Lastname TO Prefix
   100% indexed      49 records indexed
. LIST Phone, Lastname
Record#  Phone           Lastname
     16  (555)241-1108   Kotky
     34  (555)243-6331   Loftus
     20  (555)244-4526   Rivera
     37  (555)244-5922   Keegan
     38  (555)244-5922   Keegan
                 .
                 .
                 .
     21  (555)990-7647   Orlando
```

If you want an alphabetical list of all of the Agents, use the UNIQUE option:

```
. INDEX ON Agent TO Uagent UNIQUE
  100% indexed       4 records indexed
. LIST Agent
Record#  Agent
      2  JT
      5  LB
      1  MM
      3  RA

. USE
```

## See Also

CLOSE, FIND, REINDEX, SEEK, SET INDEX, SET UNIQUE, USE

## INPUT

INPUT is primarily used in dBASE programs to prompt a user to enter an expression from the keyboard. Data entry is terminated by a ↵.

## Syntax

INPUT [ < prompt > ] TO < memvar >

## Usage

The prompt you specify is a character expression. If the prompt is a character string rather than a character memory variable, the prompt must be delimited by single quotes (′), double quotes (″), or square brackets([ ]).

If a memory variable is used as a prompt, it must have been defined previously, and must not be enclosed in quotes.

Any valid dBASE III PLUS expression can be entered in response to the INPUT command. The type of expression entered determines the type of memory variable created. For instance, if you enter a number in response, a numeric memory variable is created.

If an invalid expression is entered in response to the INPUT command, an error message appears.

You must enter a response followed by ↵. If you press ↵ without first making an entry, the prompt is redisplayed.

## Programming Notes

If it is important that the response be character type data, use ACCEPT, WAIT, or @...GET with the PICTURE option. ACCEPT assumes that all user response is character and doesn't require delimiters.

If you use an INPUT command that requires a date response, include instructions in the prompt as to how the date should be entered. Entry of the CTOD( ) function or other processing of the entry is required when typing in date information.

## Examples

Character and numeric data can be placed in memory variables using prompts such as these:

```
. INPUT 'What is your last name: ' TO mname
What is your last name? 'Buckman'
. INPUT 'What is your age? ' TO mage
What is your age? 30
. DISPLAY MEMORY
MNAME        pub   C  'Buckman'
MAGE         pub   N            30  (       30.00000000)
    2 variables defined,        18 bytes used
  254 variables defined,      5982 bytes available
```

## See Also

ACCEPT, WAIT

## INSERT

INSERT adds a single new record to the database file at the current record location.

## Syntax

INSERT [BLANK] [BEFORE]

## Usage

INSERT displays the new record for full-screen data entry. Data may be entered into this record only. The new record is inserted immediately after the current record. For instance, if the current record is number 5, INSERT creates a new record 6; the old record 6 becomes record 7, and so on.

In two instances, INSERT works like APPEND: if the file is indexed, the record is placed at the end of the file; and if a record is inserted at the end of a file, it adds multiple records, one at a time.

Enter data into a memo field by placing the cursor on it (labeled **memo** on the screen) and pressing **Ctrl-PgDn**. To exit the memo field, press **Ctrl-End** to save, or **Esc** to exit without saving changes. While editing the memo field, the control keys are the same as those used with MODIFY COMMAND. Press **F1** to toggle on and off the menu displaying the keys you can use.

Arrow keys move the cursor within a record. **Esc** abandons the process without inserting a new record, displaying the message **Record is not inserted**. **Ctrl-End** terminates the process and completes the record insertion.

## Options

The BEFORE clause inserts a new record just before the current record. For instance, if the current record is number 5, INSERT BEFORE creates a new record 5; the old record 5 becomes record 6, and so on.

If BLANK is included, a new record is INSERTed, but full-screen mode is not entered. Data may be added later using one of the editing commands.

## Tip

To copy contents of the preceding record to the INSERTed record, use SET CARRY ON.

## Example

The Clients database file from the Sample Programs and Utilities disk is used for the following example.

To insert a new record immediately before record 4 (that is, to create a new record 4), type the following:

```
. SET STATUS ON
. USE Clients
. GO 4
. INSERT BEFORE
```

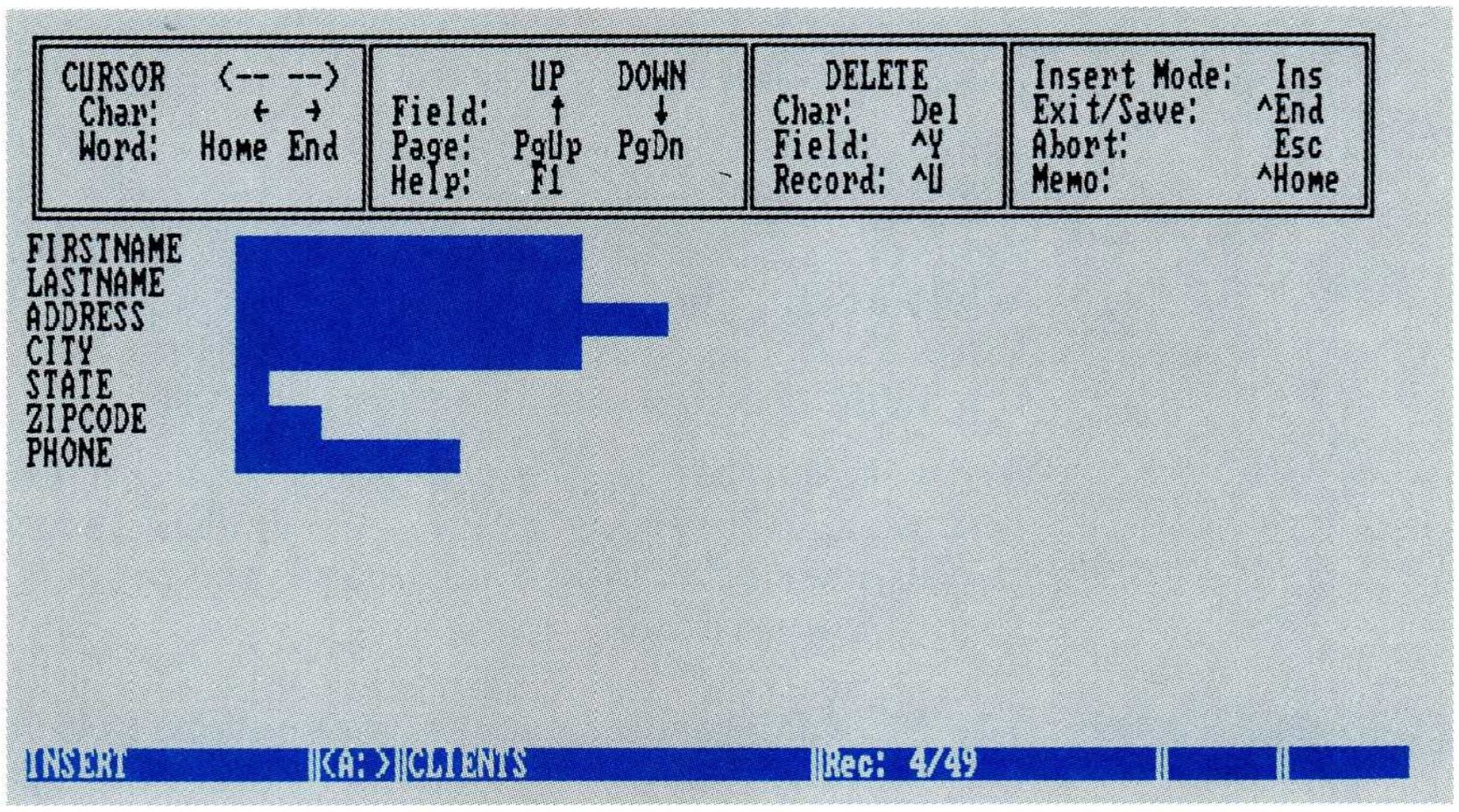



Figure 5-13 The INSERT command

## See Also

@, APPEND, CHANGE, EDIT, MODIFY COMMAND, SET CARRY, SET FORMAT

## JOIN

JOIN creates a new database file by merging specified records and fields from two open database files.

## Syntax

JOIN WITH <alias> TO <filename> FOR <condition>
[FIELDS <field list>]

## Defaults

The TO filename must include the drive and directory location if the file is not in the current directory. The file extension .dbf is assumed unless otherwise specified.

## Usage

When JOINing an active file with an open file from an unselected work area, the second file is identified by its alias name. This may be the same as its filename.

The field list may consist of any type of field from both files except memo fields. If you try to join memo fields, you will get the message **Operation with memo field invalid**.

If no field list is specified, field assignments are made from the active file first. Then, fields are assigned from the second file until the 128-field limit is reached. Duplicate field names do not appear in the new file.

JOIN updates a catalog if open and SET CATALOG is ON.

## Record Pointer

The record pointer is set to the first record in the active file. Each record in the second file is evaluated to see if it matches the FOR <condition>. If the specified condition is True, a new record is added to the new file. When all records in the second file are scanned, the record pointer in the active file advances one record, and the process is repeated. This continues until all records in the active file are processed. This operation can take a long time for large files.

## Tips

Do not use the single letters A through J or M as database filenames, because they are reserved for ALIAS names. For example, AA is a valid database filename whereas A is not.

Make sure you have enough room when JOINing two files. It is possible for two database files to be JOINed such that the new file exceeds the available disk space. It is possible for the new JOINed file to become quite large if the specified condition is not chosen carefully. For example, if two 1,000-record files were JOINed and the specified condition was always True, one million records would be created.

## Special Cases

If both files have a field name in common and you want to specify a field from the unselected file, specify the field in the following form: alias->field name. Fields which are specified using the SET FIELDS TO command may also be used. Use only the field name to specify a field in the active database file.

## Examples

You can combine the Travel.dbf database file with the Clients.dbf database file and form a new database file, Newlist, to include first and last names, addresses, and the cost of trips.

```
. SELECT 2
. USE Clients

. SELECT 1
. USE Travel
```

In the following JOIN command example, Newlist is the new .dbf file created with the fields Firstname, Lastname, City, and Cost from the two database files, Travel.dbf and Clients.dbf. Because Address and City are fields found only in Clients, which is in the unselected work area, you have to let dBASE III PLUS know where to find these fields. You can identify fields in work area 1 by the alias A or Travel, or work area 2 by the alias B or clients.

```
. JOIN WITH B TO Newlist FOR Lastname=B->Lastname FIELDS Firstname,
Lastname, B->Address, B->City, Cost
      53 records joined
. USE Newlist
. LIST STRUCTURE
Structure for database: B:Newlist.dbf
Number of data records:      53
Date of last update    : 08/02/85
Field  Field Name   Type       Width    Dec
    1  FIRSTNAME    Character    20
    2  LASTNAME     Character    20
    3  ADDRESS      Character    25
    4  CITY         Character    20
    5  COST         Numeric      10      2
** Total **                     96

. LIST NEXT 5 Lastname,Cost
Record#  Lastname                 Cost
      1  Buckman               1199.00
      2  Lisbonn               1378.00
      3  Bicksby               3195.00
      4  Garnett               1969.00
      5  Kafmanan               988.00
```

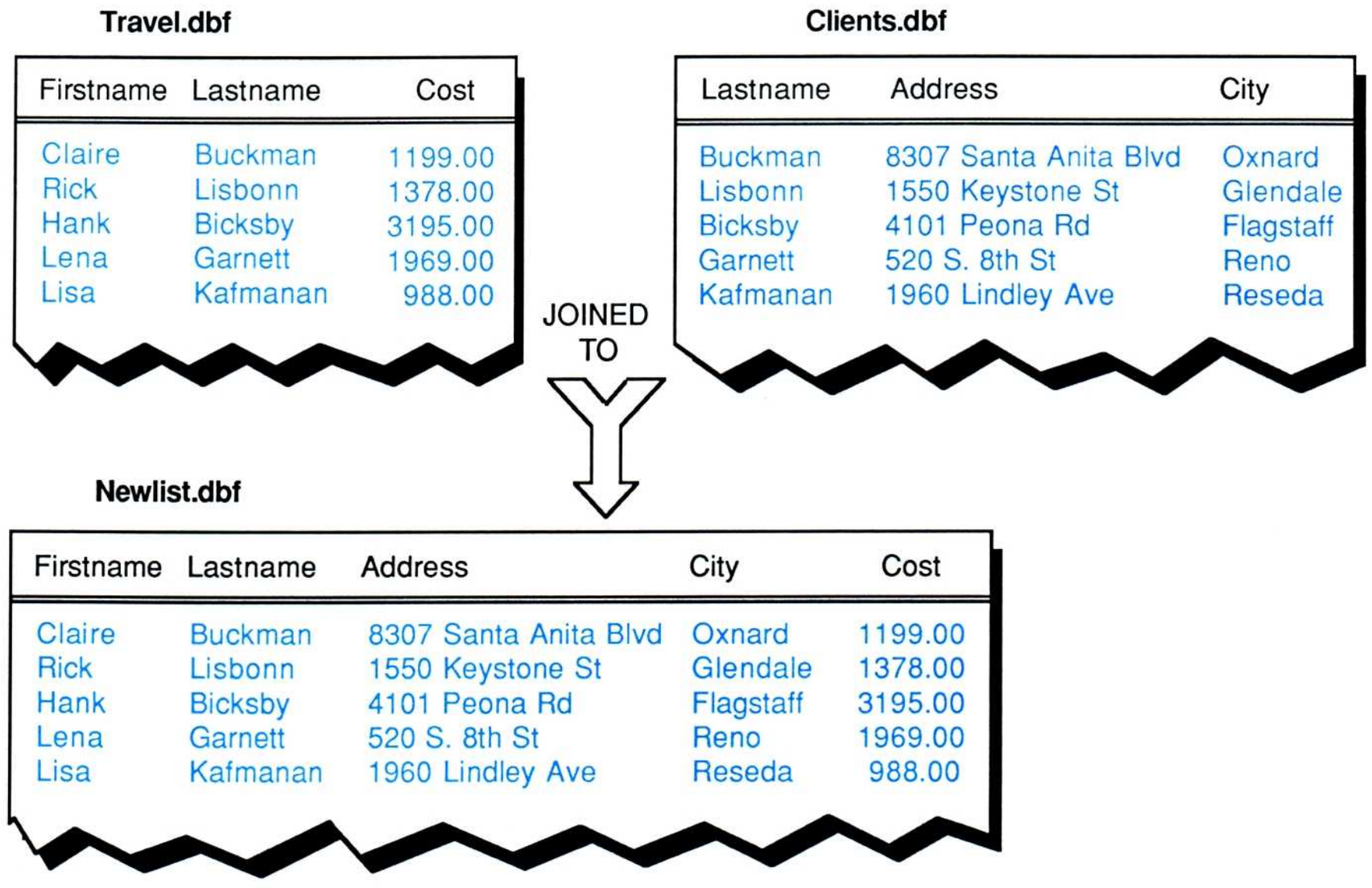

Travel.dbf
Firstname Lastname Cost
Claire Buckman 1199.00
Rick Lisbonn 1378.00
Hank Bicksby 3195.00
Lena Garnett 1969.00
Lisa Kafmanan 988.00
JOINED TO
Clients.dbf
Lastname Address City
Buckman 8307 Santa Anita Blvd Oxnard
Lisbonn 1550 Keystone St Glendale
Bicksby 4101 Peona Rd Flagstaff
Garnett 520 S. 8th St Reno
Kafmanan 1960 Lindley Ave Reseda
Newlist.dbf
Firstname Lastname Address City Cost
Claire Buckman 8307 Santa Anita Blvd Oxnard 1199.00
Rick Lisbonn 1550 Keystone St Glendale 1378.00
Hank Bicksby 4101 Peona Rd Flagstaff 3195.00
Lena Garnett 520 S. 8th St Reno 1969.00
Lisa Kafmanan 1960 Lindley Ave Reseda 988.00

Figure 5-14 The JOIN command

## See Also

SET FIELDS, SET RELATION

## LABEL FORM

LABEL FORM uses a specified label format file designed with MODIFY LABEL to print or display labels.

### Syntax

LABEL FORM < label filename > /? [ < scope > ] [SAMPLE]
[WHILE < condition > ] [FOR < condition > ]
[TO PRINT/TO FILE < filename > ]

### Defaults

File names must include the drive designator if the file is not on the default drive, unless a path is specified. If no file extension is specified in the label filename, dBASE III PLUS assumes an .lbl extension.

If no file extension is specified in the TO FILE filename, dBASE III PLUS assumes a .txt extension. Unless otherwise specified by the scope, the WHILE or FOR clause, or an active filter, labels are printed for all records in the active database file.

### Options

Use the SAMPLE option to print test labels to ensure proper alignment of the labels in the printer. The process may be repeated as many times as necessary by responding Y when asked **Do you want more samples?** When you respond with N, labels begin printing.

The TO FILE option is used to send the labels to a standard ASCII text file.

Use the catalog query clause to activate a menu of all label files for the active database file.

### Tip

When printing labels, make sure that the labels are in the printer and the printer is turned on before issuing the command. If the printer is not on, the command will not be executed immediately. After a short pause, the labels will be displayed on your screen. If you realize your printer is not on after you issue the command and you turn it on immediately, the labels will begin printing.

### See Also

CREATE/MODIFY LABEL

## LIST

LIST is used to view the contents of a database file.

### Syntax

LIST [OFF] [ < scope > ] [ < expression list > ]
[WHILE < condition > ] [FOR < condition > ]
[TO PRINT]

### Defaults

All records are displayed unless limited by the scope or a FOR or WHILE clause.

The OFF option suppresses the record numbers.

### Usage

LIST and DISPLAY ALL are nearly identical, except that LIST does not pause periodically.

To halt a LIST, press **Ctrl**-S (with SET ESCAPE ON) . Press any key to continue.

To abandon a LIST, press **Esc** (with SET ESCAPE ON).

### Example

```
. USE Tours
. GOTO 25
. LIST OFF REST Travelcode, Travelplan, Departure, Unitcost, Percent
Travelcode Travelplan                  Departure Unitcost Percent
WA8        8-night Waikiki             11/23/85    814.00   6.000
FR9        9-night Frankfurt           11/25/85   1442.00   6.000
JH10       10-night Japan/Hong Kong    11/23/85   1779.00   7.555
CH22       22-day China Majestic       11/21/85   3495.00   7.500
PC14       14-day Panama Canal Cruise  08/15/85   1595.00   6.500
HK11       11-night Hong Kong Tour     08/23/85   1299.00  10.000
. USE
```

### See Also

DISPLAY, SET HEADING, SET MARGIN

# LIST HISTORY

## LIST HISTORY

LIST HISTORY outputs a list of commands that have been executed and are stored in the history buffer. It scrolls without pausing at each full-screen display.

## Syntax

LIST HISTORY [LAST < expN > ] [TO PRINT]

## See Also

DISPLAY HISTORY

# LIST MEMORY



## LIST MEMORY

LIST MEMORY shows the name, type, and size of each memory variable. It also shows the number of active memory variables, the number of additional variables possible, the amount of memory used, and the amount of memory still available for memory variables.

### Syntax

LIST MEMORY [TO PRINT]

### See Also

DISPLAY MEMORY

# LIST STATUS

LIST STATUS provides information about the current dBASE III PLUS session. The same information is displayed as with the DISPLAY STATUS command, except that this information scrolls without pausing at each full-screen display.

## Syntax

LIST STATUS [TO PRINT]

## Usage

For each open database file, LIST STATUS shows the following information:

- Database name with drive and directory location
- Work area number and Alias
- Database relations
- Open index filenames and index key expressions
- Open memo filenames
- Filter formulas
- Format files

It also shows:

- Loaded modules
- ON command settings
- Currently selected work area
- Current file search path
- Default disk drive
- Print destination
- Currently open PROCEDURE file
- Current settings for most ON/OFF SET commands
- The setting for DEVICE (SCREEN or PRINT)
- Left margin setting
- Function key assignments

## See Also

DISPLAY STATUS

# LIST STRUCTURE



## LIST STRUCTURE

LIST STRUCTURE provides the following information about the active database: the filename, the number of data records, the last date that any database item was changed, the complete definition of each of the fields, and the total number of bytes in a record. A > symbol appears beside each field specified with the SET FIELDS TO command if SET FIELDS is ON. This information scrolls without pausing at each full-screen display.

**Syntax**

LIST STRUCTURE [TO PRINT]

**See Also**

DISPLAY STRUCTURE, SET FIELDS

## LOAD

LOAD allows you to load binary program files in memory and run them directly from memory with the CALL command.

## Syntax

LOAD <binary filename>

## Usage

LOAD places a binary (.bin) file in memory where it can be executed with the CALL command.

## Programming Notes

dBASE III PLUS treats each loaded file as a subroutine or program module, not as an external dBASE program file.

A maximum of sixteen files may be resident in memory at one time. Each can be up to 32,000 bytes and must be a binary type file.

Each LOADed module must have a unique name. The default extension is .bin. When you CALL a file, omit the extension; the filename itself becomes the module name.

If you LOAD a file that has the same filename but a different extension than one already LOADed, the new one replaces the first one in memory.

To see the names of LOADed modules, issue the DISPLAY/LIST STATUS command.

dBASE III PLUS does not check the integrity of the files you LOAD. Make sure the programs are in binary form and each program executes properly.

You must conform to the following specifications while designing the assembly language program from which the binary file is made:

- It must originate (ORG) the first executable instruction at an offset of zero.
- The program must not allocate or use memory over and above its actual size, because LOAD uses the file size to determine how much memory to allocate.
- The assembly language program must not lengthen or shorten memory variables passed as arguments from CALL...WITH in DS:BX.
- Before returning control to dBASE III PLUS, the program must restore both the Code Segment (CS) and the Stack Segment (SS) Registers.
- Return control to dBASE III PLUS using a far return (RET FAR). Most commercial programs end with an exit call rather than a far return; execute these with the RUN/! command, not with LOAD and CALL.

To prepare an assembly language program written in 8086/8088 assembler for LOADing by dBASE III PLUS, use the DOS macro assembler and the following DOS program commands:

- To assemble programs and create an object (.obj) file:
  MASM <filename> <filename> NULL NULL
- To link OBJ files and create an executable (.exe) file:
  LINK <filename> NULL NULL
- To create a binary (.bin) file from an executable file:
  EXE2BIN <filename>

## Examples

These steps take you through the process of LOADing and running the following assembly language program, named Cursor.

```
;= = = =
```

Use the following program to execute Cursor from within dBASE III PLUS:

```
* Place file into memory.
LOAD Cursor
*
* Define shape of cursor as a block instead of
* an underline, and change its shape...
Shape = CHR(18)
CALL Cursor WITH Shape
* CALL Cursor WITH CHR(18) would also work.
*
* Remove LOADed module from memory.
RELEASE MODULE Cursor
```

## See Also

CALL, RELEASE

## LOCATE

LOCATE searches the active database file for a record that satisfies specified criteria.

### Syntax

LOCATE [ < scope > ] [WHILE < condition > ]
[FOR < condition > ]

### Defaults

Unless otherwise restricted by the scope or a WHILE clause, LOCATE searches the entire database beginning with the first record.

### Usage

The NEXT < n > option for the scope limits the search to the specified number of records and begins the search at the current record.

To find the next occurrences of the specified condition, use the CONTINUE command.

### Record Pointer

If a match is found, the record pointer moves to that record and displays the record number. FOUND( ) returns .T.

If no match is found, the record pointer moves to the end of the file (EOF( ) is .T.), or the end of the scope if one was specified. dBASE III PLUS displays the message **End of locate scope**. FOUND( ) returns .F.

### Special Cases

Unless the scope is NEXT < n > , REST, or WHILE, LOCATE starts searching at the beginning of the file. If there are multiple LOCATEs, the most recent overrides any other.

LOCATE and CONTINUE are specific to the work area in which they are issued. You can have a different LOCATE/CONTINUE for each work area. If you leave the work area, they will still be in effect when you return.

### Examples

To locate the first record in Travel.dbf that contains the Travelcode HI14 or CN7:

```
. USE Travel
. LOCATE FOR Travelcode = 'HI14' .OR. Travelcode = 'CN7'
Record =        5
```

To LOCATE the first of the next five records that contains XX in the Travelcode field:

```
. GO TOP
. LOCATE NEXT 5 FOR Travelcode = 'XX'
End of LOCATE scope
. ? RECNO()
         5
. USE
```

## See Also

CONTINUE, FOUND()

# MODIFY COMMAND

## MODIFY COMMAND

MODIFY COMMAND is the dBASE III PLUS full-screen text editor. Its main function is to create and edit dBASE program and format files, but it can also be used to create or edit any standard ASCII text file.

## Syntax

MODIFY COMMAND/FILE <filename>

## Defaults

If the drive, directory, or extension are not explicitly specified as part of the filename with MODIFY COMMAND, the default drive and directory and the .prg extension are used. MODIFY FILE is an alternative syntax for the dBASE text editor which does not provide a default extension. If no extension is specified when you create a file, none is provided.

The maximum file size that MODIFY COMMAND/FILE can handle is 5,000 bytes, unless an external word processor is specified in a dBASE III PLUS Config.db file. (Refer to Chapter 4 for instructions on setting up an external word processor as the default dBASE III PLUS text editor.) A warning message is issued if an attempt is made to exceed the 5,000-byte limit, and the text in the file may not be reliable after it is saved.

## Usage

When you issue the MODIFY COMMAND/FILE command, dBASE III PLUS searches for the specified file. If the file exists, it is called up for editing. Each time a file is edited, the previous version is saved as a backup file with the .bak extension. If the file is not found, a new file is created.

When you first begin MODIFY COMMAND/FILE, the keys you can use appear in a menu at the top of the screen. Pressing **F1** toggles this menu on or off.

You can use the following keys:

| Keys | Action |
|---|---|
| ← → ↑ ↓ | Moves the cursor one position in the indicated direction |
| **Backspace** | Deletes the character to the left of the cursor |
| **F1**, **Ctrl**-\ | Toggles the cursor menu on and off |
| **Ctrl**-A, **Home** | Moves the cursor to the beginning of the current or the previous word |
| **Ctrl**-B, **Ctrl**-→ | Moves the cursor to the end of the line |
| **Ctrl**-C, **PgDn** | Scrolls the screen forward 18 lines (or less if cursor navigation menu is displayed) |
| **Ctrl**-D | Equivalent to the → key |
| **Ctrl**-E | Equivalent to the ↑ key |
| **Ctrl**-F, **End** | Moves the cursor to the beginning of the next word |
| **Ctrl**-G, **Del** | Deletes the character at the cursor |
| **Ctrl**-KB | Reformats the paragraph |
| **Ctrl**-KF | Finds the first occurrence of a specified string |
| **Ctrl**-KL | Finds the next occurrence of the string found with **Ctrl**-KF |
| **Ctrl**-KR | Reads another file into the file being edited |
| **Ctrl**-KW | Writes the entire file to another file |
| **Ctrl**-M, ↵ | Moves the cursor to the beginning of the next line. If INSERT is ON, inserts a blank line. |
| **Ctrl**-N | Inserts a blank line after the cursor |
| **Ctrl**-Q, **Esc** | Exits without saving changes |
| **Ctrl**-R, **PgUp** | Scrolls the screen back 18 lines (or less if cursor navigation menu is displayed) |
| **Ctrl**-S | Equivalent to the ← key |
| **Ctrl**-T | Deletes all characters from the current cursor position to the beginning of the next word |
| **Ctrl**-V, **Ins** | Toggles between the insert and overwrite modes. Insert mode stays on until toggled OFF, or until the current dBASE III PLUS session ends. |
| **Ctrl**-W, **Ctrl-End** | Exits and saves changes |
| **Ctrl**-X | Equivalent to the ↓ key |
| **Ctrl**-Y | Deletes the current line |
| **Ctrl**-Z, **Ctrl**-← | Moves the cursor to the beginning of the line |

## Notes

When you type a line that is longer than 66 characters, MODIFY COMMAND continues it on the next line. You may also type a semicolon (;) at the end of a line to indicate that a command continues on the next line. This allows you to break long command lines to make your program file more readable. dBASE III PLUS will recognize that the continued line is part of the same command line when you try to run the program file.

Although you cannot enter a line that exceeds 66 characters, you may insert additional characters into existing text, to a maximum of 79 characters per line. This enables you to read in files with lines 79 characters wide.

To get a printout of a file created by MODIFY COMMAND, issue the command TYPE <filename> TO PRINT at the dot prompt.

You cannot edit memo fields with MODIFY COMMAND. Use the EDIT or CHANGE command.

## See Also

RENAME, TYPE

**MODIFY LABEL** See CREATE/MODIFY LABEL.

**MODIFY QUERY** See CREATE/MODIFY QUERY.

**MODIFY REPORT** See CREATE/MODIFY REPORT.

**MODIFY SCREEN** See CREATE/MODIFY SCREEN.

**MODIFY STRUCTURE** See CREATE < newfile > /MODIFY STRUCTURE.

**MODIFY VIEW** See CREATE/MODIFY VIEW.

## NOTE/*/&&

NOTE, an asterisk (*), or double ampersand (&&) characters are used to indicate comment lines.

### Syntax

NOTE/* <text> and
[<command>]&& <text>

### Usage

If a NOTE/* line ends with a semicolon, dBASE III PLUS reads the next line as part of the comment line.

To insert comments on the same line as a command, use && after the command and before the comment. You may also use && at the beginning of a line.

### Examples

This example shows how you might use NOTE/*.

```
NOTE This is a simple loop
STORE 1 to x
DO WHILE x < 100
   STORE x+1 TO x
ENDDO
```

The following example illustrates the use of && to put a comment on a program line:

```
memvar = 12      && initializes numeric memvar
```

### See Also

MODIFY COMMAND

## ON

ON branches the execution of a program if one of three specified conditions occurs.

### Syntax

ON ERROR/ESCAPE/KEY <command>

### Usage

You can specify a command to be executed for each of three ON conditions. The ON conditions remain in effect until you explicitly remove them, or until you leave dBASE III PLUS.

To remove an ON, enter ON ERROR, ON ESCAPE, or ON KEY without specifying a command.

Note that ON ESCAPE does not work when SET ESCAPE is OFF.

dBASE III PLUS responds to the ON options in the following order of precedence:

| | | |
|---|---|---|
| ON ERROR | = | A dBASE III PLUS error occurred |
| ON ESCAPE | = | The **Esc** key is pressed |
| ON KEY | = | Any key is pressed |

If, for example, ON KEY and ON ESCAPE are in effect at the same time, pressing the **Esc** key executes the ON ESCAPE command. It does not branch to the ON KEY command line.

If only ON KEY is in effect and SET ESCAPE is OFF, pressing the **Esc** key activates the command specified in the ON KEY command line.

If the ON KEY command is in effect, and any key other than **Esc** is pressed, the ON KEY command is executed after the current command is completed. For instance, pressing a key while INDEXing a file will not interrupt the index procedure. However, it is important to note that unless the program uses the stored key, the command specified on the ON KEY command line will execute repeatedly.

Proper use of ON KEY requires that the key pressed and stored in the buffer is removed from the buffer with commands such as INKEY() or READ. If you use ON KEY WAIT TO <memvar>, the key that was pressed to cause the program to branch to the ON KEY command is put into the memory variable; you do not have to press a key for this to occur.

ON ERROR does not respond to errors at the operating system level, such as a drive or printer not being accessible. It responds only to dBASE III PLUS errors, such as a syntax error or a nonexistent file. Also, returning from an ON ERROR < DO program > will not re-evaluate an IF or DO WHILE condition in the program in which the error occurred.

## Examples

The ON examples below assume that the procedure file containing the two procedures IF_err and Stop is already open (see the SET PROCEDURE TO command).

To activate the ON ERROR option, for example, to execute the DO < program > command when a dBASE III PLUS error occurs, type the following:

```
ON ERROR DO If_err
```

Then, if dBASE III PLUS encounters an error, it branches to:

```
PROCEDURE If_err
 <commands>
.
.
* Turn off the on error flag if you
* don't want to trap additional errors.
ON ERROR
RETURN
```

The next example shows you how to set up the ON KEY option to execute a DO < program > command. To halt printing and branch to a procedure named Stop, enter:

```
ON KEY DO Stop
```

Afterward, if you press any key, the program branches to the Stop procedure. The Stop procedure prompts you to press the letter A if you want to stop printing; pressing any other key would cause printing to resume.

```
PROCEDURE Stop
* First clear the key that triggered ON KEY
* from the type-ahead buffer with INKEY().
i = INKEY()
WAIT "Press A to Abort printing, " +;
     "any other key to continue" TO choice
IF choice $ "Aa"
   RETURN TO MASTER
ENDIF
RETURN
```

In this example, the substring operator $ evaluates the key pressed in response to the WAIT command to see if it is either an upper-case or lower-case letter A.

## See Also

INKEY(), READKEY(), SET ESCAPE

## PACK

PACK removes records that are marked for deletion from the active database file.

## Syntax

PACK

## Usage

All open index files are automatically REINDEXed.

After you execute a PACK command, the disk space used by the deleted records is reclaimed when the file is closed.

## See Also

DELETE, RECALL, REINDEX, ZAP

## PARAMETERS

PARAMETERS assigns local variable names to data items passed from a calling program. This command receives variables passed using the DO <filename> WITH <parameter list> command.

## Syntax

PARAMETERS <parameter list>

## Usage

When included in a dBASE program file, PARAMETERS must be the first executable command. In a procedure file, the PROCEDURE name is the first command, followed by the PARAMETERS command. The parameters you pass can be any legitimate expressions. The parameter list assigns local variable names to receive parameters from the sending program's parameter list. The local variables created by specifying PARAMETERS are discarded when control is returned to the calling program. The parameter list must contain the same number of items as in the calling program.

If the parameter being passed is a memory variable, its value may be changed. The change is transferred to the variable in the calling program. If the parameter being passed is an expression, the value of that expression is stored to a memory variable in the receiving program.

## Examples

The following command file calculates the area for a given length and width.

```
*The command file Areacalc.prg
PARAMETERS Length,Width,A
*A corresponds to area
A=Length * Width
RETURN
*EOF:  Areacalc.prg
```

To execute this command file:

```
. area = 0
0
. DO Areacalc WITH 6,4, area
                    24
. ? area
      24
```

## See Also

DO, PRIVATE, PROCEDURE, PUBLIC, SET PROCEDURE

## PRIVATE

PRIVATE allows you to create new memory variables in a lower-level program with the same names as memory variables that were created in a calling program or were previously declared as PUBLIC.

### Syntax

PRIVATE <memvar list> /[ALL [LIKE/EXCEPT <skeleton>]]

### Usage

Changes made to a PRIVATE memory variable do not overwrite the values in memory variables with the same name in other programs. Previous values of PRIVATE memory variables are retained in memory with the name of the programs that created them.

When the program containing PRIVATE memory variables ends, the values that were hidden by PRIVATE are reinstated. The PRIVATE values are no longer in effect.

### See Also

DO, PARAMETERS, PUBLIC

## PROCEDURE

PROCEDURE identifies the beginning of each routine in a procedure file.

### Syntax

PROCEDURE <procedure name>

### Usage

Each separate procedure within a procedure file must begin with the PROCEDURE command followed by the procedure name.

When dBASE III PLUS encounters a PROCEDURE <procedure name> statement in a command file that is not part of a larger SET PROCEDURE TO <filename>, it immediately executes a RETURN.

Procedure names may be up to 8 characters long. They may contain letters, numbers, and underscores, and must begin with a letter.

### Examples

The following file contains two routines which are used to output specific messages to the screen.

```
* This is an example of the procedure file Proc1.prg

PROCEDURE Message1
   @ 15,0 CLEAR
   @ 18,0 SAY 'The account is now overdrawn'
RETURN
PROCEDURE Message2
   @ 10,0 CLEAR
   @ 15,10 SAY 'Payment has not been received'
   @ 17,10 SAY 'Please issue account hold'
RETURN
```

To activate a procedure within Proc1.prg:

```
SET PROCEDURE TO Proc1
DO Message1
```

### See Also

SET PROCEDURE TO

## PUBLIC

PUBLIC makes memory variables global by enabling them to be used and altered by any dBASE program or subprogram. Unlike other memory variables, they are not released when the program ends.

## Syntax

PUBLIC < memory variable list >

## Usage

Memory variables which you want to be PUBLIC must be declared PUBLIC before they are given values. The values of PUBLIC memory variables may be changed by any program that uses them.

The value of a PUBLIC memory variable may be hidden temporarily when a program or procedure file is called, by declaring memory variables with the same name PRIVATE, or passing a parameter with the DO WITH < parameter list > command.

Variables saved in a memory file are always RESTOREd as PUBLIC variables, if they're restored at the dot prompt. Variables restored from a memory file in a dBASE program are RESTOREd as PRIVATE variables unless you specify otherwise. To RESTORE PUBLIC variables in a dBASE program file, you first need to redeclare the variables you intend to be PUBLIC as PUBLIC variables. Then RESTORE FROM < filename > ADDITIVE restores the variables as PUBLIC, overwriting any variables of the same name.

**NOTE**
Variables you declare PUBLIC are stored as logical type variables until you initialize them.

## Example

For example, within a dBASE program file, the Page and Answer memory variables are declared PUBLIC when the following commands are executed:

```
PUBLIC Page, Answer
Page = 1
Answer = 'Y'
```

## See Also

DO, PARAMETERS, PRIVATE

## QUIT

QUIT closes all open files, terminates the dBASE III PLUS session, and returns control to the operating system.

### Syntax

QUIT

### Usage

This is the only safe method for exiting from dBASE III PLUS after executing commands. Exiting dBASE III PLUS by resetting your computer may damage open files and cause data loss.

## READ

READ activates all @...GETs issued since the last CLEAR, CLEAR ALL, CLEAR GETS, or READ. It is most commonly used in dBASE program files for full-screen entry or editing of data.

### Syntax

READ [SAVE]

### Usage

The READ command uses the same full-screen editing keys as the APPEND and EDIT commands. You need to use the READ command to edit memory variables with the @ command.

READ clears all GETs after you finish editing.

### Options

READ SAVE does not clear GETs. This means that the next time you issue READ, the same set of GETs appears for editing. If you use the READ SAVE option, make sure to issue CLEAR GETS before you reach the 128 @...GET command maximum or any user-defined GETs maximum specified in a dBASE III PLUS Config.db file. Refer to Chapter 4 for more information.

### Programming Notes

If you want to use a multi-page format (.fmt) file in which the @...SAY...GETs continue on from 2 to 32 screen pages, include a READ wherever you want a page break.

### Example

Use the READ command to edit the contents of a memory variable:

```
STORE SPACE(25) TO mname
@ 10,10 SAY "Enter your name: " GET mname
READ
```

READ puts the cursor into the Mname variable to allow editing. The SPACE(25) function initializes a blank character string that is 25 characters long.

### See Also

CLEAR, CLEAR GETS, CLEAR MEMORY, CREATE/MODIFY SCREEN, SET DEVICE TO, SET FORMAT TO

## RECALL

RECALL reinstates records that are marked for deletion in the active database file.

## Syntax

RECALL [ < scope > ] [WHILE < condition > ]
[FOR < condition > ]

## Defaults

Unless otherwise specified by the scope or the WHILE or FOR clause, only the current record is RECALLed.

## Special Cases

RECALL does not reinstate records that have been removed from the database by the PACK or ZAP commands.

RECALL ALL has no effect with SET DELETED ON. If SET DELETED is ON, you must specify which records to RECALL.

## Examples

The Clients database file from the Sample Program and Utilities disk is used for the following examples.

To reinstate only the first record in the database, assuming that records 1, 5, and 10 are marked for deletion with DELETE:

```
. USE Clients
. RECALL
      1 record recalled
```

To reinstate record 10:

```
. RECALL RECORD 10
      1 record recalled
```

To reinstate record 5 and any other records that have been marked for deletion:

```
. RECALL ALL
      1 record recalled
```

## See Also

DELETE, DELETED(), PACK, SET DELETED ON, ZAP

## REINDEX

REINDEX rebuilds all active index (.ndx) files.

### Syntax

REINDEX

### Usage

Whenever you REINDEX an index (.ndx) file that was created with SET UNIQUE ON or with UNIQUE included in the INDEX syntax, the rebuilt index file retains its UNIQUE status, regardless of whether UNIQUE is ON or OFF.

### Example

The Clients database file from the Sample Programs and Utilities disk is used to illustrate this command.

```
. USE Clients
. SET INDEX TO Location,Cnames
. REINDEX
Rebuilding index - B:Location.ndx
    49 Records indexed
Rebuilding index - B:Cnames.ndx
    49 Records indexed
```

### See Also

INDEX, PACK, SET INDEX TO, SET UNIQUE, USE

## RELEASE

RELEASE deletes memory variables and thus opens memory space for other use or is used to remove a LOADed assembly language program from memory.

## Syntax

RELEASE <memvar list> /[ALL[LIKE/EXCEPT <skeleton>]]/ [MODULE <module name>]

## Options

In the skeleton, a question mark (?) masks a single character, and an asterisk (*) masks one or more characters.

In the interactive mode, RELEASE ALL deletes all memory variables. Within a program, it only deletes PRIVATE memory variables created in the current program. PRIVATE variables created in higher-level programs are not released by RELEASE ALL.

RELEASE MODULE removes a LOADed module from memory. This allows you to RELEASE LOADed assembly language program modules from memory.

## Examples

To RELEASE all memory variables starting with the letter m:

```
. RESTORE FROM Mem
. RELEASE ALL LIKE m*
. DISPLAY MEMORY
PGX1        pub  N    1  (          1.00000000)
PGX2        pub  N    3  (          3.00000000)
PAGES       pub  N   33  (         33.00000000)
    3 variables defined,        27 bytes used
  253 variables available,    5973 bytes available
```

To RELEASE all memory variables except those that have the letter x as the third character:

```
. RESTORE FROM Mem
. RELEASE ALL EXCEPT ??x*
. DISPLAY MEMORY
PGX1        pub  N    1  (          1.00000000)
PGX2        pub  N    3  (          3.00000000)
      2 variables defined,          18 bytes used
    254 variables available,      5982 bytes available
```

To RELEASE the Mvar, Pages, Pgx1, and Pgx2 variables:

```
. RESTORE FROM Mem
. RELEASE Mvar, Pages, Pgx1, Pgx2
. DISPLAY MEMORY
MEM1        pub  C   "Clients"
   1 variables defined,         9 bytes used
 255 variables available,     5991 bytes available
```

## See Also

CALL, CLEAR MEMORY, LOAD, RESTORE, RETURN, SAVE, STORE

## RENAME

RENAME changes the name of a disk file.

### Syntax

RENAME <old filename> TO <new filename>

### Defaults

Both the old and new filenames must include file extensions. If the files are not on the default drive, precede the filenames with drive designations.

### Usage

The new filename cannot be an existing file in the same directory.

If you rename a database file with an associated memo file, you must also rename the corresponding memo (.dbt) file to match.

An open file cannot be RENAMEd.

### Tips

Do not use the letters A through J or M as database filenames because they are reserved for ALIAS names. However, you can specify AA as a valid database filename.

### Examples

To RENAME a database file named School.dbf to Roster.dbf:

```
. RENAME School.dbf TO Roster.dbf
```

To RENAME the file Letters.jim to Memos.old, assuming that the file resides on the B drive and that the default drive is currently A:

```
. RENAME B:Letters.jim TO B:Memos.old
```

### See Also

CLOSE, USE

## REPLACE

REPLACE alters the contents of specified fields in the active database file.

## Syntax

REPLACE [ < scope > ] < field > WITH < exp > [, < field > WITH < exp > ... ] [WHILE < condition > ] [FOR < condition > ]

## Defaults

Unless otherwise specified by the scope or the WHILE or FOR clause, only the current record is replaced.

## Usage

The field and the WITH expression must have the same data type. With numeric fields, the WITH expression may not be larger than the field width. If it is, asterisks replace the field contents, and some data may be lost.

**NOTE**
Replacements on a key field of an indexed database file also update the index file if it's in use. When the replacement is made, the record moves to a new position in the index file. If you specify a scope, WHILE, or FOR when making replacements on an indexed field, not all of the records you intended will be replaced. For example, if you REPLACE ALL, only the first record and those that follow the new key field value will be replaced. That's because each record is reindexed as the REPLACE is done, and the record pointer is moved to follow the replaced record. To REPLACE all the records, you should use the database file without an index, and then REINDEX afterwards.

## Record Pointer

The REPLACE command does not reposition the record pointer unless a scope, FOR, or WHILE is specified. If the record pointer is at the end of the file (EOF( ) is .T.), no replacements are made unless a SCOPE or FOR condition is specified in the command. Be sure to position the pointer with a command such as GO TOP before beginning a REPLACE option.

## Examples

The following examples use the original version of the Tours database file from the Sample Programs and Utilities disk. (If you have modified this file, your results may be slightly different.)

```
. USE Tours
. REPLACE Percent WITH 9.5 FOR Percent = 5.0
      6 Records Replaced
```

To raise the travel plan cost on all plans by ten percent:

```
. REPLACE ALL Unitcost WITH Unitcost * 1.1,Percent WITH Percent * 1.05
     30 Records Replaced
```

## REPORT

REPORT prints information from the active database using the report form file created by CREATE or MODIFY REPORT. A report may be directed to the screen, the printer, or an ASCII text file.

### Syntax

REPORT FORM < filename > /? [ < scope > ]
[WHILE < condition > ] [FOR < condition > ] [PLAIN]
[HEADING < expC > ] [NOEJECT]
[TO PRINT/TO FILE < filename > ] [SUMMARY]

### Defaults

The default extension is .frm for the FORM filename.

Unless otherwise specified by the scope, a FOR clause, or an active filter condition or query file, all records in the database are included in the report.

### Usage

If group subtotals are defined in the report form file, the active database should be INDEXed or SORTed on the same fields as in the group expression.

### Options

Use the catalog query clause to activate a menu of all report forms for the active database file.

PLAIN causes the report to print without page numbers or a system date. The page heading defined in the report form file appears on the first page only.

HEADING, followed by a character expression, defines an extra heading that is printed on the first line of each page. If the heading is a character string, you must delimit it. PLAIN and HEADING are mutually exclusive, with PLAIN taking precedence.

NOEJECT, with TO PRINT, suppresses the usual initial form feed, causing the report to print on the first page that comes up in the printer.

The semicolon (;) character in a character or memo field causes a carriage return and line feed in a report form. This semicolon does not print.

TO FILE sends the report to a text (.txt) file on the default drive (and displays it on the screen), unless otherwise directed.

SUMMARY suppresses display of detail lines, showing only subtotals and totals on the report.

## See Also

CREATE/MODIFY REPORT

## RESTORE

RESTORE retrieves and activates memory variables from a memory file.

### Syntax

RESTORE FROM <filename> [ADDITIVE]

### Defaults

A .mem file extension is assumed unless otherwise specified.

### Usage

When you RESTORE memory variables in a dBASE program file, all of the currently active memory variables are deleted unless ADDITIVE is specified. Unless you redeclare memory variables as PUBLIC in a dBASE program file, the variables are restored as PRIVATE. Memory variables you restore from the dot prompt, however, always come back as PUBLIC.

No more than 256 memory variables may be active at any time. dBASE III PLUS allocates 6,000 bytes for active memory variables (unless you specify a different amount MVARSIZ in the dBASE Config.db file). To find out how to change this limit, see Chapter 4.

### Options

To retain the current memory variables, include the ADDITIVE option, which causes the RESTOREd memory variables to be added to the current ones. Any current memory variables with the same name are overwritten. To restore variables in a memory file as PUBLIC, you must redeclare them as PUBLIC before you RESTORE the memory file and use the ADDITIVE option.

### Examples

The memory file Mem.mem is on the Sample Programs and Utilities disk.

```
. DISPLAY MEMORY
    0 variables defined ,        0 bytes used
  256 variables available,  6000 bytes available
.RESTORE FROM Mem
. DISPLAY MEMORY
MVAR        pub   N             33 (          33.00000000)
MEM1        pub   C          "Clients"
PGX1        pub   N              1 (           1.00000000)
PGX2        pub   N              3 (           3.00000000)
PAGES       pub   N             33 (          33.00000000)
         5 variables defined,        45 bytes used
       251 variables available,   5955 bytes available
```

To RESTORE the variable Mvar from the memory file Mem as PUBLIC in a dBASE program file:

```
PUBLIC mvar
RESTORE FROM mem ADDITIVE
```

## See Also

PUBLIC, RELEASE, SAVE, STORE

## RESUME

RESUME works in conjunction with SUSPEND. It causes a SUSPENDed program to continue execution.

### Syntax

RESUME

### Usage

When you issue the RESUME command, the previously SUSPENDed command file restarts execution at the command line immediately following the line on which you stopped the file.

### Tips

It is good practice to issue the CLEAR command before you issue RESUME. This ensures that the commands you entered while the program was SUSPENDed do not intefere with subsequent screen output.

### See Also

CANCEL, RETURN, SUSPEND

## RETRY

RETRY is used mainly in error recovery. RETRY returns control from a program in progress to the prior program from which it was called. The command in the first program that called the second program is then re-executed. This is in contrast to the RETURN command, which executes the next line.

## Syntax

RETRY

## Usage

The primary use of RETRY is in error recovery situations where it can repeat a command until it is successfully executed. Note, however, that RETRY will not re-evaluate an IF or DO WHILE condition.

When executing a program called with ON ERROR, RETRY returns control to the program where the error originated, and re-executes the command in which the error occurred.

RETRY can be used in procedures as well as dBASE program files. When RETRY is used in a program file, it closes the file. RETRY clears out the value returned by the ERROR( ) function.

## Examples

The following command files illustrate how to recover from the **File is already open** message with an ERROR( ) number of 3:

```
* Main.prg
ON ERROR DO RECOVER WITH ERROR()
USE My_file
* EOF:Main.prg
```

```
* Recover.prg
PARAMETERS Error
IF Error=3
   CLOSE DATABASES
ENDIF
RETRY
* EOF:Recover.prg
```

## See Also

ERROR(), RETURN

## RETURN

RETURN is used in programs to restore control to calling programs, or to the dot prompt. When control is restored, the command following the calling command is executed.

### Syntax

RETURN [TO MASTER]

### Usage

If the TO MASTER option is used in a program, control returns to the highest-level calling program.

RETURN can be used in procedures as well as command files. If used in a command file, RETURN closes the file.

If the RETURN command is not included in a program, dBASE III PLUS exits and closes the program after it executes the last command in the file, or when it reaches a CANCEL or QUIT command. It returns to the program that called it or, if it is executed in the highest-level program, to the dot prompt.

RETURN releases all PRIVATE memory variables but does not affect PUBLIC variables. RETURN also clears out any value returned by the ERROR( ) function.

### See Also

ERROR( ), PRIVATE, PUBLIC

## RUN/! < command >

RUN executes a specified DOS command, or any program which can be executed by DOS, from within dBASE III PLUS.

## Syntax

RUN/! < command >

## Usage

RUN executes the specified DOS command or program. When execution is finished, control returns to dBASE III PLUS.

RUN requires more memory than the 256K needed to run dBASE III PLUS. It requires enough additional memory for the DOS Command.com file, plus the amount required for the command or program file itself. If your computer does not have sufficient memory for the RUN command, dBASE III PLUS returns the message **Insufficient memory**.

The operating system file Command.com must be in the current directory, or specified by the DOS COMSPEC parameter.

Use the DOS SET COMSPEC command to tell the operating system where to find Command.com. For example, SET COMSPEC = A:\Command.com. Refer to your DOS manual for more information on SET and COMSPEC.

**WARNING**
Some DOS commands, such as ASSIGN and PRINT, remain in memory after being RUN, occupying memory that should be restored to dBASE III PLUS. You should run these commands before loading dBASE III PLUS. If you are not able to load dBASE III PLUS or get an **Insufficient memory** error message, clear any memory-resident programs or restart your computer and try again. Also, do not run commands which may not leave enough memory for dBASE III PLUS to operate.

## Example

To reset the system date from a memory variable

```
. Today = '7/5/85'
7/5/85
. RUN DATE &Today
. ? DATE()
07/05/85
```

## SAVE

SAVE stores all or part of the current set of memory variables to a disk file.

## Syntax

SAVE TO <filename> [ALL LIKE/EXCEPT <skeleton>]

## Defaults

The filename must include the drive designator if the file is not on the default drive. Unless otherwise specified, the file extension for any file created with this command is .mem.

If the ALL LIKE/EXCEPT option is not used, all current memory variables are saved to the disk file.

## Usage

In the skeleton, a question mark (?) masks a single character and an asterisk (*) masks one or more characters.

## Tip

SAVE does not overwrite an existing file with the same name as the TO file without asking, unless SET SAFETY is turned OFF.

## Examples

To SAVE all memory variables to the disk file Myfile.mem located on disk drive B:

```
. SAVE TO B:Myfile
```

To SAVE all memory variables with names beginning with the letter d to the existing disk file Myfile.mem located on disk drive B:

```
. SAVE ALL LIKE d* TO B:Myfile
Myfile.mem already exists, overwrite it? (Y/N)
```

To SAVE all memory variables except those with names containing the letter "x" as the fourth character to the disk file Myfile.old on the default drive:

```
. SAVE ALL EXCEPT ???X* TO Myfile.old
```

## See Also

RESTORE, SET SAFETY, STORE

## SEEK

SEEK searches for the first record in an indexed database file with a key that matches the result of a specified expression. SEEK conducts a very rapid record search.

## Syntax

SEEK < expression >

## Usage

The specified expression must be a valid dBASE III PLUS expression and must have the same data type as the index key expression.

Substring or partial key searches work only if the search expression matches the index key, starting with the leftmost character. The criteria for the search also depend on the setting of SET EXACT.

For example, if the index key is 'Smith, John' with SET EXACT OFF, 'Smi' will find the index key. However, if SET EXACT is ON, the search expression would have to match the full length of the index key, which in this example would be 'Smith, John'.

## Record Pointer

If the search is unsuccessful, the message **No find.** is displayed, and the record pointer moves to the end of the file (EOF( ) is .T.). Use SET TALK OFF to suppress the message.

The FOUND( ) function is set to True (.T.) when the SEEK is successful.

## Example

The Tours database file and its index Tour date on the Sample Programs and Utilities disk are used to demonstrate the command.

To SEEK a date < expression >:

```
. USE Tours INDEX Tourdate
. SEEK CTOD('11/03/85')
. ? Seats_open, Departure
   32   11/03/85
```

## See Also

EOF( ), FIND, FOUND( ), INDEX, SET DELETED, SET EXACT, SET INDEX, USE

## SELECT

SELECT chooses a work area from the ten which are available to open a database file or selects a a work area in which a database file is already open.

## Syntax

SELECT < work area/alias >

## Usage

When you enter dBASE III PLUS, the active work area is work area one. The valid work areas are 1 through 10, or A through J or M. You can open a database file in each work area you select. Work area 10 is reserved for an open catalog. You cannot open a database file in this area if a catalog is open. If you have a database file open in work area 10 and then open a catalog, your database file will be closed automatically since the catalog has precedence.

Each work area is also used for opening other files associated with the database file, for example, index, query, and format files. However, the same associated files cannot be open in more than one work area at the same time.

You can use SELECT to change work areas so that the database file open in the selected work area becomes the active database file. You can change the current work area by specifying the alias of the database file open in another work area. If you do not specify an alias when you open the file, the filename (without the extension) is the default alias name.

To use a variable to select a work area or its alias, the variable must be used as a macro (&) function. For example, if the alias name is stored in a character memory variable, type SELECT & < memvar > .

Normally, commands display or change the contents of fields in the active database file. However, you can specify fields in other work areas containing related files using the syntax:

alias- > field name

If you have issued the SET FIELDS TO command, and SET FIELDS is ON, you can also display or change data for the fields in the specified FIELDS list in other work areas.

## Record Pointer

Each work area maintains a separate record pointer. Moving between work areas does not affect the position of any record pointer. Commands that change the position of a record pointer affect only the active database file, unless you set a relation with SET RELATION TO, or you open a view file in which a relation is set.

## Tips

A maximum of 15 open files of all types is allowed. If you get the error message **Too many files are open** and the number of open files is less than 15, verify that the Config.sys file on your startup disk or hard disk root directory specifies FILES = 20 and BUFFERS = 15 (refer to your computer's operating system manual).

Work area 10 is reserved for a catalog file if one is open.

## Examples

In this example, the Clients database file is opened in work area 1, and the Travel database file is opened in work area 3.

```
. SELECT 1
. USE Clients ALIAS Names
. SELECT 3
. USE Travel
```

To change the work area, SELECT the alias of the file or the number or letter of the work area.

```
. SELECT Names
```

In these examples, work areas were selected by using a work area number, and the alias name of a database file open in a work area. You could also select a work area by a letter, for example, selecting work area 1 by entering SELECT A.

## See Also

SET CATALOG, SET FIELDS, SET VIEW

## SET

SET is a full-screen, menu-driven command for displaying and changing the values of many SET commands.

## Syntax

SET

## Usage

To exit the SET menu, press **Esc**. This retains any changes made to a SET command, but only for the duration of your current session in dBASE III PLUS. You can alter the default value of SET commands by changing the value in the dBASE Config.db file. Refer to Chapter 4 for more information.

For details on any SET command, see the individual entry of the same name in this chapter, for example, SET CONSOLE.

## Options

SET provides seven choices: **Options**, **Screen**, **Keys**, **Disk**, **Files**, **Margin**, and **Decimals**.

### Options

The Options Menu contains SET commands with two possible values, such as **Help**, which can be ON or OFF, and **Device**, which can be TO SCREEN or TO PRINT.

### Screen

The Screen Menu allows you to change the color and screen attributes.

### Keys

The Keys Menu lets you reprogram the function keys on your keyboard. **F1** is reserved for use as a Help key and is not programmable. From the Keys Menu, you can define the **F2** through **F10** keys for commands of up to 33 characters. (You can set a command of up to 238 characters with the SET FUNCTION command.)

### Disk

The Disk Menu lets you specify a default disk drive and a drive search path.

### Files

The Files Menu opens Alternate, Format, and Index files. When you choose *Alternate*, you enter the name of the Alternate file. Choosing any of the others activates a menu list of possible files, such as all .fmt files in the open catalog, or on the default disk drive if no catalog is open.

**Margin** The Margin Menu sets the left margin in reports and the width of displays in memo fields.

**Decimals** The Decimals Menu specifies the maximum number of decimal places displayed as the result of a numeric function or calculation.

## SET ALTERNATE

SET ALTERNATE records all output other than that of full-screen commands to a text file.

### Syntax

SET ALTERNATE ON/OFF

SET ALTERNATE TO [ < filename > ]

### Defaults

SET ALTERNATE is normally OFF.

The filename must include the drive designator if the file is not on the default drive. The file extension .txt is assumed unless otherwise specified.

SET ALTERNATE TO ↵ closes an open ALTERNATE file. CLOSE ALTERNATE is an alternative syntax.

### Usage

This command consists of two parts:

SET ALTERNATE TO < filename > creates a text file and opens the file. It overwrites any file with the same name.

SET ALTERNATE ON starts recording of keyboard entries and screen displays in the alternate file. Full-screen operations such as @...SAY, EDIT, and APPEND are not recorded.

You can use SET ALTERNATE OFF to suspend writing to the text file without closing the text file. You can then start recording again with SET ALTERNATE ON. The text file will not be closed until you issue the CLOSE ALTERNATE command. Be sure you issue the CLOSE ALTERNATE command before using the text file (for example, with your word processor). The file created by SET ALTERNATE is a standard ASCII text file and may be edited with most word processing programs.

## Examples

The commands below illustrate the SET ALTERNATE commands. To create the alternate text file, type:

```
. SET ALTERNATE TO Textfile
```

Commands executed after you open an alternate file are not recorded until you SET ALTERNATE ON. Type:

```
. SET ALTERNATE ON
    <commands - recorded>
```

All dBASE commands executed are stored in the ALTERNATE file until you close the ALTERNATE file or suspend recording of commands.

To suspend recording of commands, type:

```
. SET ALTERNATE OFF
      <more commands - not recorded>
```

To resume writing to the text file, type:

```
. SET ALTERNATE ON        <commands>
      <commands>
```

To close the text file, type:

```
. CLOSE ALTERNATE
```

## See Also

CLOSE

## SET BELL

SET BELL determines whether a warning bell rings when you enter an invalid data type in an entry field or reach the end of the input area.

### Syntax

SET BELL ON/OFF

### Defaults

SET BELL is normally ON.

The warning bell rings unless SET BELL is turned OFF.

## SET CARRY

SET CARRY copies data from the prior record to a new record when using APPEND or INSERT.

### Syntax

SET CARRY ON/OFF

### Defaults

SET CARRY is normally OFF.

The command does not affect INSERT BLANK or APPEND BLANK; a blank record is still added.

### Usage

Used with the APPEND command, the last record is carried over into the appended record. Pressing **PgDn** will carry over the same record contents into each appended record. Pressing **Ctrl**-W will save the record and exit the APPEND command. Simply pressing ↵ will not save the record.

With INSERT BEFORE, the preceding record's contents are carried over into the inserted record. For example, if you type GO 4 followed by INSERT BEFORE, the new record has the same contents as record 3.

### See Also

@, APPEND, INSERT, READ, SET FORMAT

## SET CATALOG

The SET CATALOG command is used to create or open a catalog file. When a catalog is open and SET CATALOG is ON, new files which are used are added to the catalog.

### Syntax

SET CATALOG ON/OFF
SET CATALOG TO [ <filename> ]/?

### Defaults

SET CATALOG is normally OFF, and no catalogs are selected. However, selecting a catalog with SET CATALOG TO automatically SETs CATALOG ON.

### Usage

SET CATALOG TO <filename> is used to open an existing catalog or, if one doesn't exist, to create a new catalog. New catalog filenames are limited to eight characters. Catalog filenames are always given the .cat file extension.

When you create a new catalog, you are prompted to enter a one-line text description of the catalog file (provided SET TITLE is ON). This file description will be displayed later, if you use the SET CATALOG TO ? command to select from a list of available catalogs. The description for each catalog entry is displayed when the catalog name is highlighted.

A master catalog file, Catalog.cat, is created the first time you create a catalog. dBASE III PLUS always checks to see if this main catalog exists. If it doesn't, dBASE III PLUS creates one. The master catalog is stored in the current directory and keeps track of other catalogs. You can create different master catalogs in other drives or directories.

The master catalog allows you to use the SET CATALOG TO ? command. It stores catalog filenames along with the catalog title descriptions.

Catalogs are database files which have a predefined file structure. When you create or select a catalog file, it is automatically opened in work area 10. When a catalog is open, work area 10 is reserved for the system to update the catalog file, so you cannot select and use that work area for another database file.

# SET CATALOG



When you open or create a new catalog, SET CATALOG is automatically set ON. From then on, all database files (and associated files such as index, query, report, and label files) you use, or new files you create, are added to the catalog. With SET CATALOG ON, the open catalog records entries for files associated with open database files, and it is updated automatically by certain commands such as CREATE, INDEX, and CREATE/MODIFY REPORT. You will be prompted to enter a file title description for any file you create or add while SET CATALOG is ON (if SET TITLE is ON).

Whenever you open a catalog, dBASE III PLUS checks the catalog contents against the disk. If you've previously deleted any files with SET CATALOG OFF, dBASE III PLUS updates the catalog by deleting the appropriate filename to match the disk.

If you decide you want to close a catalog, you use the command, SET CATALOG TO ↵. If you just want to stop adding files to an open catalog, you can SET CATALOG OFF. Then, when you want to resume adding files to the catalog, SET CATALOG ON again.

SET CATALOG OFF, unlike turning off the catalog with SET CATALOG TO ↵, keeps the catalog open and allows you to use the query clause (?) with other dBASE commands. For example, with SET CATALOG OFF, you can type MODIFY REPORT ?, and dBASE III PLUS activates a menu list of .frm files in the open catalog. Because the files in the catalog are linked with the database file they were created or used with, only the files relevant to the active database file are listed.

**NOTE**
If SET CATALOG is OFF when a database file is used and is later set ON, files you use with the open database file (for example, an index file) are not added to the catalog. dBASE III PLUS displays the message **File not cataloged, since SET CATALOG was OFF when the active database file was USEd.**

## Catalog Structure

All catalogs have the same file structure. With the exception of the Title and Tag fields, dBASE III PLUS automatically fills in the field contents whenever a new file is entered in the catalog. The structure is:

| *Field* | *Field Name* | *Type* | *Width* | *Dec* |
|---|---|---|---|---|
| 1 | PATH | Character | 70 | |
| 2 | FILE_NAME | Character | 12 | |
| 3 | ALIAS | Character | 8 | |
| 4 | TYPE | Character | 3 | |
| 5 | TITLE | Character | 80 | |
| 6 | CODE | Numeric | 3 | |
| 7 | TAG | Character | 4 | |
| ** Total ** | | | 181 | |

There are instances in which a field is intentionally left blank. For example, because only database files have ALIAS names, the ALIAS field is only used to define database (.dbf) files.

**Path** This tells dBASE III PLUS which path to search if the called catalog is not on the default drive and path. The path is stored exactly as it is entered, with the filename.

**File_name** This is the name of the file, including its extension.

**Alias** If you do not assign an alias name, dBASE III PLUS treats the database file filename as the alias name. This field is left blank for all other file types.

**Type** This field contains the default file extension, identifying the type of file. Even if you specified a different extension when creating the file, the default extension is entered in the Type field. However, the extension that you specified is included in the File_name field.

**Title** Filling in this field is optional. dBASE III PLUS prompts you to add a description when you create the catalog (if SET TITLE is ON). You have up to 80 spaces to describe the contents of the new catalog. If you want to change the contents and the catalog is not already in USE, type:

```
. USE <catalog name>.cat
. EDIT
```

If the catalog *is* in USE:

```
. SELECT 10
. EDIT
```

**Code** When you have a catalog open, Code is the number dBASE III PLUS assigns to each database file put into USE. Associated index (.ndx), format (.fmt), label (.lbl), query (.qry), report form (.frm), and screen (.scr) files are also assigned the same number. View (.vue) files are not assigned a number, however, since they store relations or links between database files.

**Tag** This field is not used.

## Adding Entries

A new entry is added automatically to the active catalog when you use any of the following commands:

COPY STRUCTURE
COPY STRUCTURE EXTENDED
COPY TO
CREATE
CREATE FROM
CREATE/MODIFY LABEL
CREATE/MODIFY QUERY
CREATE/MODIFY REPORT
CREATE/MODIFY SCREEN
CREATE/MODIFY VIEW
IMPORT FROM
INDEX
JOIN
SET FILTER TO FILE
SET FORMAT
SET VIEW
SORT
TOTAL
USE

If the filename already exists in the catalog, you are prompted for a file title (if SET TITLE is ON). Each of these commands also allows you to select files using the **?** query from the currently open catalog, regardless of whether SET CATALOG is ON or OFF.

**NOTE**
Use the SET TITLE OFF command to suppress the file title prompt, if you do not want to enter file titles. To prevent the catalog from being updated without closing the catalog, use the SET CATALOG OFF command.

## Example

You can use the Triplog.cat catalog file on the Sample Programs and Utilities disk for this example. First open the catalog:

```
. SET CATALOG TO Triplog
. USE ?
```

Select the Travel.dbf file from the database file menu list, then type:

```
. SET INDEX TO ?
```

Only the files which index the Travel database file appear. Select the Trips.ndx file. Now, to select a report file and begin printing, type:

```
. REPORT FORM ? TO PRINT
```

dBASE III PLUS displays a submenu of reports which are related to the currently selected database file. If you select a report form file, the report will then be printed.

## See Also

SET TITLE

## SET CENTURY

SET CENTURY allows the input and display of century prefixes in the year portion of dates.

### Syntax

SET CENTURY ON/OFF

### Defaults

SET CENTURY is normally OFF, meaning that the century is not displayed and can't be input.

The default date entry and display is six digits, allowing only two digits for a year entry (current twentieth century only). However, if a date calculation results in a non-twentieth century date, this value can be placed into a date field, and it will retain the correct century. If the field is edited with a full-screen editing command and SET CENTURY is OFF, the year is changed to a twentieth-century date.

Although the four-digit year displayed with SET CENTURY ON makes the date display ten characters, the database field size is always eight bytes. This is because dates are stored internally as numbers. Also, the format in which a date is displayed does not affect its internal storage.

### Usage

With SET CENTURY ON, the date display contains a four-digit year. When SET CENTURY is OFF, the year is displayed in two digits, with the twentieth century assumed.

SET CENTURY affects all full-screen editing commands and date displays. When SET CENTURY is OFF, you can input only twentieth century dates.

### Examples

```
. ? DATE()
02/14/85
. SET CENTURY ON
. ? DATE()
02/14/1985
. SET CENTURY OFF
```

### See Also

CTOD(), DATE(), DTOC(), YEAR()

## SET COLOR

SET COLOR allows the selection of colors for use on color monitors, or screen attributes for use on monochrome monitors.

### Syntax

SET COLOR ON/OFF

SET COLOR TO [[ <standard> ] [, <enhanced> ] [, <border> ] [, <background> ]]

where standard and enhanced settings specify foreground and background colors separated by /.

### Defaults

SET COLOR ON/OFF switches between color and monochrome monitors on systems equipped with both. The default is whatever the system is using when you start dBASE III PLUS.

SET COLOR TO ↵ resets the screen to dBASE III PLUS default colors, not the Config.db settings. The default settings are:

COLOR — standard: white on black
background: black on white
MONO — standard: dim
background: dim, dim reverse

### Usage

Use the SET COLOR TO command to select the colors to display text, background, border, and highlighted (enhanced) areas on your screen. An example of these areas is shown below.

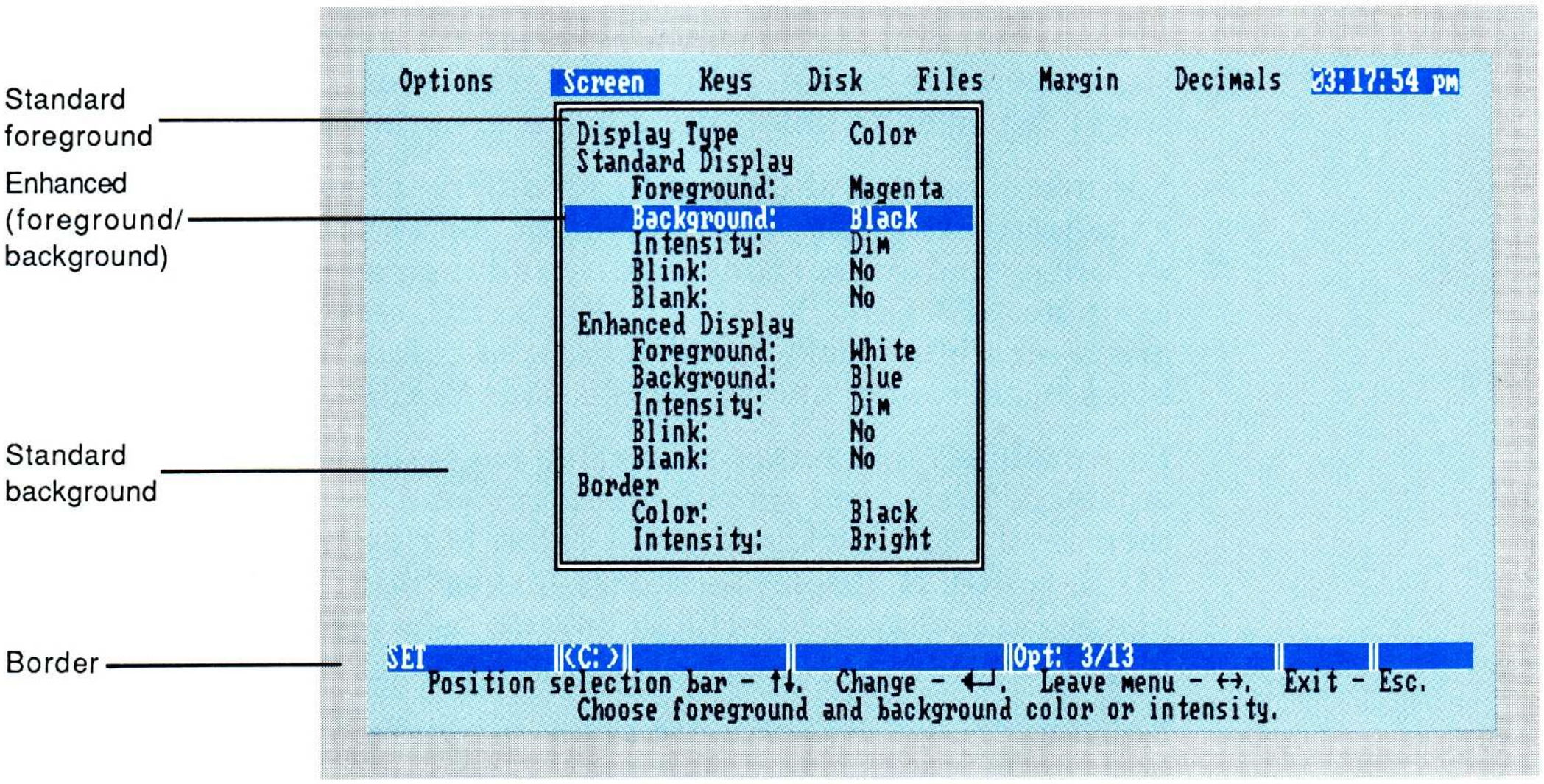



Figure 5-15 Screen display colors

On a color or monochrome monitor, the standard display is white letters on a black background. The border area is also black. Enhanced areas, for example, the status bar on The Assistant Menu, appear in reverse, with black letters on a white background.

You can change the color settings by specifying letter codes for each of the types of displayed text: standard, enhanced, border, and background. The letters used to select colors are as follows:

| Color | Letter Code | Color | Letter Code |
|---|---|---|---|
| Black | N or blank | Red | R |
| Blue | B | Magenta | RB |
| Green | G | Brown | GR |
| Cyan | BG | Yellow | GR + |
| Blank | X | White | W |
| Gray | N + | | |

Table 5-4 Color table

You may also specify an asterisk (*) following a selected color to indicate blinking, or specify a plus sign ( + ) to indicate high intensity (except for brown, yellow, and grey colors). When high intensity is displayed in color, the color is generally brighter and lighter.

For machines that allow you to set different colors for standard and enhanced text areas, set the *standard* and *enhanced* colors in pairs as foreground/background colors, and then specify the border. For example, SET COLOR TO G/B, RB/N, BG, sets *standard* colors to green on a blue background, *enhanced* colors to magenta on a black background, and provides a cyan border.

For machines that require the same background for standard and enhanced areas, you specify the foreground colors individually, and then set the [, < background > ] color. For example, SET COLOR TO B, G, RB, R, displays standard text as blue over red standard display areas, enhanced (highlighted) text as green on red, with a magenta border.

You can change one or more color attributes at the same time. The position of commas indicates the setting you want to change. For example, to set the border to green without affecting the other options, you would type SET COLOR TO ,,G. In specifying foreground/background pairs, you can omit the foreground or background color. When you do not specify a color, either before or after the slash [/] in the foreground/background pair, the color black is selected.

If you do not want characters in a specific area to appear on your screen (for instance, for a password entry), specify Blank (X) for either a foreground or background color in those areas.

You can store a color setup in a dBASE Config.db file, however, the syntax is then:

COLOR = < settings >

**WARNING**
Color settings in dBASE programs created for a color monitor will have an unpredictable effect when used on a monochrome monitor.

For a monochrome monitor, you can specify the letters U for underline and I for inverse video instead of using colors to provide emphasis for standard and enhanced foreground text. If you do specify colors, or the letters U or I, make sure they work properly for the particular machine and application you are running.

The background option is not available on all machines. It works only on machines that cannot set the background colors of individual characters with the standard or enhanced options. On these machines, <background> sets the color for the entire background.

## Special Cases

Notice to Compaq users: Compaq computers function as though they have color monitors. Thus, the monochrome attributes U and I do not generate underlining and inverse video. Also, on the Compaq, color settings are translated to shades of black and white on your screen.

## Examples

To set standard video with yellow letters on a red background, enhanced (highlighted) video with white letters on a red background, and the border to yellow, type the following:

```
. SET COLOR TO GR+/R,W/R,GR+
```

If standard and enhanced colors have the same background, the highlighted areas will not be visible. You should change one of the background colors.

To set these colors on machines that can't set individual character backgrounds, type the following:

```
. SET COLOR TO GR+, W, GR+, R
```

To set colors for the enhanced display only, for example, to blue over a high intensity yellow background with a red border, type:

```
. SET COLOR TO ,B/GR+,R
```

To set the color to use the default black color on both standard and enhanced text areas, with a red border:

```
. SET COLOR TO W/,/W,R
```

## See Also

SET, SET INTENSITY

## SET CONFIRM

SET CONFIRM is used in full-screen editing to determine whether the cursor moves automatically to the next entry field when the end of the current one is reached, or if you must first press ↵.

### Syntax

SET CONFIRM ON/OFF

### Defaults

SET CONFIRM is normally OFF.

### Usage

SET CONFIRM OFF causes the cursor to advance from one entry field to the next when the first entry field is filled.

SET CONFIRM ON causes the cursor to remain at the last space in an entry field until ↵ is pressed.

### See Also

SET BELL

## SET CONSOLE

SET CONSOLE turns the screen display on and off from within a program.

### Syntax

SET CONSOLE ON/OFF

### Defaults

SET CONSOLE is normally ON.

### Usage

SET CONSOLE affects only the output intended for the screen. Output routed to the printer is unaffected. SET CONSOLE works only within a program. You use SET CONSOLE OFF to prevent the output of commands from affecting the display on your screen (for example, if you are printing reports routed to your printer which you don't want to display on your screen). From the dot prompt, SET CONSOLE is always ON.

Keyboard input requested by commands such as WAIT and ACCEPT is accepted, although the prompts from these commands are not visible, and the user will not see what is typed. However, @...SAY...GETs override the CONSOLE setting and are visible regardless of the SET CONSOLE status.

Error messages are not suppressed by SET CONSOLE OFF.

## SET DATE

SET DATE determines the format for date displays.

### Syntax

SET DATE AMERICAN/ANSI/BRITISH/FRENCH/GERMAN/
ITALIAN

### Defaults

DATE is normally SET to AMERICAN.

### Usage

This command allows more flexibility in date output.

| | |
|---|---|
| AMERICAN | mm/dd/yy |
| ANSI | yy.mm.dd |
| BRITISH/FRENCH | dd/mm/yy |
| GERMAN | dd.mm.yy |
| ITALIAN | dd-mm-yy |

Some of the formats are:

```
. Date = CTOD('11/05/85')
11/05/85
.SET DATE ANSI
. ? Date
85.11.05
. SET DATE BRITISH
. ? Date
05/11/85
. SET DATE AMERICAN
. ? Date
11/05/85
```

### See Also

DATE(), SET CENTURY

## SET DEBUG

SET DEBUG is a tool for locating errors in a program. It determines whether output from SET ECHO is sent to the screen or printer.

### Syntax

SET DEBUG ON/OFF

### Defaults

SET DEBUG is normally OFF.

### Usage

When SET DEBUG is ON, the output of SET ECHO is routed to the printer. This prevents interference between the program's screen operations and the screen displays that are normally generated by the debugging process.

With both SET DEBUG ON and SET ECHO ON, the output for the TALK (for example, the number of records indexed), LIST, and DISPLAY commands, and error messages are *not* sent to the printer unless SET PRINT is ON.

### See Also

SET ECHO, SET TALK

## SET DECIMALS

SET DECIMALS determines the minimum number of decimal places that dBASE III PLUS displays for the results of numeric functions and calculations.

### Syntax

SET DECIMALS TO <numeric expression>

### Defaults

A minimum of two decimal places is normally displayed.

### Usage

SET DECIMALS applies to division, SQRT(), LOG(), EXP(), and VAL() functions. If SET FIXED is ON, SET DECIMALS TO applies to all displays, not just the functions listed here.

For calculations that do not involve multiplication or division, the number of decimal places in the result is the same as in the number with the most decimal places.

For calculations that involve multiplication or division, the number of decimal places is the sum of the number of decimal places for all numbers being multiplied, unless SET FIXED is ON. In that case, SET DECIMALS TO <numeric expression> applies to all displays.

### Examples

Compare the results of several calculations using the standard number of decimal places.

```
. ? 3/4
 0.75
. ? 3.00/4.000
     0.750
. ? SQRT(4.37)
2.09
```

and using SET DECIMALS TO:

```
. SET DECIMALS TO 4
. ? 3/4
 0.7500
. ? 3.00/4.000
     0.7500
. ? SQRT(4.37)
2.0905
```

### See Also

SET FIXED

# SET DEFAULT TO



## SET DEFAULT

SET DEFAULT TO allows the user to select the drive where all operations take place and all files are stored, unless otherwise directed.

## Syntax

SET DEFAULT TO < drive >

## Defaults

The default drive is initially the drive from which you started dBASE III PLUS.

When you change drives with SET DEFAULT TO, the default directory is the directory last specified on that drive. For example, if drive A was left in subdirecory \DATA when you left it for a different drive, when you SET DEFAULT back TO A:, you will be in subdirectory \DATA again.

SET DEFAULT does not check whether or not the named drive exists and does not actually change the drive to which you return when you QUIT dBASE III PLUS.

Refer to Chapter 4 to find out how to set the default drive in the dBASE III PLUS Config.db file.

## Example

If the Clients database file is located on disk drive B but dBASE III PLUS was started from drive A, which therefore is the default drive, typing the following command:

```
. USE Clients
```

will result in the message **File does not exist**, because Clients is not on the default drive A. To change the default drive and open the database file, type:

```
. SET DEFAULT TO B
. USE Clients
```

## See Also

SET PATH

## SET DELETED

SET DELETED determines whether records that are marked for deletion are included or ignored by other dBASE III PLUS commands.

### Syntax

SET DELETED ON/OFF

### Defaults

SET DELETED is normally OFF.

### Usage

INDEX and REINDEX always include all records regardless of the status of SET DELETED.

With SET DELETED ON, most commands do not consider deleted records part of the file. For example, dBASE III PLUS will not LOCATE or LIST deleted records. However, commands in which the RECORD or NEXT <n> scope is specified, and DISPLAY (operating on current record), DISPLAY RECORD <n>, and GO/GOTO <n>, include deleted records.

If SET DELETED is ON, RECALL ALL does not recall any records, because the deleted records are ignored.

### See Also

DELETED()

## SET DELIMITERS

SET DELIMITERS determines how field widths are indicated in the full-screen mode.

### Syntax

SET DELIMITERS ON/OFF

SET DELIMITERS TO [ < expC > /DEFAULT]

### Defaults

SET DELIMITERS is normally OFF so that entry fields are not enclosed by specified delimiter characters. While SET INTENSITY is ON, entry fields are displayed in reverse video.

SET DELIMITERS ON uses colons (:) to set off the field areas, unless another delimiter is chosen with the SET DELIMITERS TO command.

### Usage

SET DELIMITERS TO < character string > sets the characters used to mark the field area. The character string must be one or two characters. If you define one character, that character marks the beginning and the end of the field. If you define two characters, the first marks the beginning of the field and the second marks the end of the field. If you define more than two characters, only the first two are read.

SET DELIMITERS must be ON for the symbols defined by SET DELIMITERS TO to be in effect.

### Examples

To mark the beginning and the end of a field with #:

```
. SET DELIMITERS TO '#'
. SET DELIMITERS ON
```

To mark the beginning of each field with [ and the end with ]:

```
. SET DELIMITERS TO '[]'
. SET DELIMITERS ON
```

To reset the delimiters to colons:

```
. SET DELIMITERS TO DEFAULT
```

When using the @...GET command, the most recently issued SET DELIMITERS character is used. The following dBASE program file provides an example:

```
SET TALK OFF
SET DELIMITERS TO "[]"
SET DELIMITER ON
STORE 'abc' TO one
STORE '123' TO two
CLEAR
@ 10,10 GET one
READ
SET DELIMITERS TO default
@ 11,10 GET two
READ
```

## See Also

@, APPEND, CHANGE, EDIT, INSERT, READ, SET INTENSITY

## SET DEVICE

SET DEVICE determines whether @...SAY commands are routed to the screen or the printer.

### Syntax

SET DEVICE TO PRINT/SCREEN

### Defaults

The default is SET DEVICE TO SCREEN.

### Usage

With SET DEVICE TO PRINT, the output from @...SAY commands is sent to the printer. @...GET commands are ignored. An @ command that specifies coordinates of lower values than previous @ commands results in a page eject.

On some printers, @...SAYs may not appear until the print buffer is emptied. To empty the buffer, issue an EJECT command or print a blank line.

### Example

Use SET DEVICE to print an @...SAY and cause a form feed:

```
. SET DEVICE TO PRINT
. @ 10,10 SAY "This is a test"
. EJECT
```

### See Also

@, EJECT, SET FORMAT

## SET DOHISTORY

SET DOHISTORY determines whether or not commands from command files are recorded in a memory buffer as they're executed. This buffer is referred to as the *history* buffer.

### Syntax

SET DOHISTORY ON/OFF

### Defaults

SET DOHISTORY is normally OFF.

### Usage

When SET DOHISTORY is ON, commands from command files are stored in the history buffer in the order of their execution. You can then edit these commands in full-screen edit mode.

Note that SET DOHISTORY has no effect on commands you issue directly from the dot prompt. dBASE III PLUS always stores these in the history buffer unless you SET HISTORY OFF. If SET HISTORY is OFF, commands executed from dBASE program files will not be stored in the history buffer.

> **NOTE**
> Editing a command in the history buffer does not change the command in the dBASE program file. You need to edit the commands in the dBASE program file using MODIFY COMMAND.

### Tips

In conjunction with SUSPEND and DISPLAY HISTORY, SET DOHISTORY is useful for debugging programs. However, because SET DOHISTORY slows down command file execution, use it only when debugging.

### See Also

DISPLAY HISTORY, LIST HISTORY, RESUME, SET DEBUG, SET ECHO, SET HISTORY, SET STEP, SUSPEND

## SET ECHO

SET ECHO displays command lines from dBASE III PLUS programs on the screen and/or the printer as they are executed (from the dot prompt, or from a dBASE program file).

### Syntax

SET ECHO ON/OFF

### Defaults

SET ECHO is normally OFF.

### Usage

Program instructions are not normally displayed during program execution. SET ECHO ON is one of four dBASE III PLUS commands which are helpful in debugging dBASE program execution. The other three commands are SET DEBUG, SET STEP, and SET TALK.

### See Also

SET DEBUG, SET PRINT, SET STEP, SET TALK

## SET ESCAPE

SET ESCAPE determines whether pressing **Esc** halts the processing of a program or pressing ← or **Ctrl**-S halts scrolling. It also affects whether INKEY() is able to record the **Esc** key (and ← and **Ctrl**-S).

### Syntax

SET ESCAPE ON/OFF

### Defaults

SET ESCAPE is normally ON.

### Usage

With SET ESCAPE ON, if you press **Esc** while a command is being processed (for example, during a LIST operation), the processing stops. If running the command from the dot prompt, dBASE III PLUS displays the message ***** INTERRUPTED *****, and returns to the dot prompt.

If you press **Esc** while a dBASE program is running, the program halts and dBASE III PLUS displays the message:

**Called from < filename.prg >**
**Cancel, Ignore, Suspend? (C, I, or S)**

With SET ESCAPE OFF, the **Esc** key is disabled and command processing cannot be interrupted. SET ESCAPE OFF also disables the operation of **Ctrl**-S or ← when they are used to temporarily suspend scrolling.

> **NOTE**
> With SET ESCAPE OFF, program execution cannot be terminated without rebooting your computer. SET ESCAPE OFF should only be used in programs that terminate naturally. Rebooting your computer with files open may cause data loss.

### See Also

INKEY(), ON ERROR/ESCAPE/KEY, READKEY()

## SET EXACT

SET EXACT determines whether a comparison between two character strings requires the strings to be the same length.

### Syntax

SET EXACT ON/OFF

### Defaults

SET EXACT is normally OFF.

### Usage

If SET EXACT is OFF, comparisons between character strings begin with the leftmost character in each string and continue character-by-character to the end of the string on the right of the relational operator. If the two strings compare favorably up to that point, the result of the comparison is true.

If SET EXACT is ON, the comparison of characters in each string is the same, except that both character strings must be the same length for the comparison to be evaluated as true.

### Examples

Consider the two character strings 'abc' and 'abcdef'. First, compare the two character strings with SET EXACT OFF.

```
. ? 'abc' = 'abcdef'
.F.
```

If the character string on the right is longer than the character string on the left, the two strings do not match, and the result of the comparison is False.

```
? 'abcdef' = 'abc'
.T.
```

Since the character string on the right is identical to the starting characters of the longer character string on the left, the two strings are considered equal.

With SET EXACT ON, the same character strings are no longer considered equal because they do not match exactly:

```
. SET EXACT ON
. ? 'abcdef' = 'abc'
.F.
```

## SET FIELDS

SET FIELDS defines a list of fields that may be accessed in one or more files. It also determines whether that list is used as the default fields or expression list for several dBASE commands.

### Syntax

SET FIELDS ON/OFF

SET FIELDS TO [ < fields list > /ALL]

### Defaults

SET FIELDS is normally OFF. It is automatically set ON when you specify a fields list with SET FIELDS TO.

When SET FIELDS OFF (the default), all the fields of the active database are available. Fields in files open in other work areas are available for display only if you specify their alias names.

When SET FIELDS is ON, only those fields listed with the SET FIELDS TO command can be displayed or altered. The fields list can contain fields of the active database file and any other files in use.

**WARNING**
If you SET FIELDS ON without first using SET FIELDS TO, *no* fields may be accessed.

SET FIELDS TO defines a list of fields that may be accessed in one or more files. It also redefines the default field or expression list used with dBASE commands which use or display fields. The list of fields specified by SET FIELDS TO is not active unless SET FIELDS is ON. Using the SET FIELDS TO command automatically turns the SET FIELDS command ON.

SET FIELDS TO ↵ removes those fields from the list that belong to the active database file. SET FIELDS TO ALL includes all the fields of the active database file in the fields list.

SET FIELDS TO does not impose any order on the fields listed. It only limits the fields that are accessible. This command is also used with CREATE VIEW FROM ENVIRONMENT to define the fields provided in the view.

After a fields list is set, additional or consecutive SET FIELDS TO commands add fields to the current fields list. To include fields from files in other work areas, SELECT the work area of the file and use the SET FIELDS TO command, or specify the file's alias with the field name.

When SET FIELDS is ON, the LIST STRUCTURE and DISPLAY STRUCTURE commands indicate the fields which are in the currently defined fields list with the > symbol.

> **NOTE**
> SET FIELDS TO can define a list of fields from multiple files, but it does not relate those files. You must use SET RELATION to establish a connection between files in different work areas.

### Other Commands Affected

SET FIELDS TO affects the potential of several other commands. Among these commands are the ones allowing you to specify FIELDS as part of their syntax. SET FIELDS TO allows these commands to support multiple fields from multiple files. The commands affected are:

- AVERAGE
- BROWSE
- CHANGE
- COPY TO
- COPY STRUCTURE
- DISPLAY
- EDIT
- JOIN
- LIST
- SUM
- TOTAL

Prior to using SET FIELDS TO, the default field or expression list for these commands is defined as all the fields in the active database file structure. SET FIELDS TO, however, redefines the list as fields that are either part of the active database file structure or part of the structure of a related file.

There are two commands that operate only on fields and can also access work areas beyond the area currently selected. These commands are @...GET and REPLACE. These commands are only able to access those fields included in the SET FIELDS list, regardless of whether an alias is specified.

APPEND and INSERT are also affected by the SET FIELDS TO commands. View files and relations established between files with SET RELATION should be used in conjunction with SET FIELDS TO only when the following information is understood:

- APPEND and INSERT add records to the active database file only, not to all database files in the view files.
- APPEND and INSERT use all fields in the active database file unless a format file is used with APPEND to exclude certain fields.
- Even though a view may be in effect, and its fields list contains fields from related files, APPEND and INSERT do not add data to fields in these files.

It is possible to perform an APPEND BLANK or INSERT BLANK in several work areas, open a format file with multiple fields from multiple files, and then use READ to append to these files simultaneously. However, this can cause problems if the files are related, and the files related to the active database file contain records with blank key fields.

Using SET RELATION, if more than one record in the related file has a key matching that of the record in the parent file, the record pointer is moved to the first matched record it finds. Therefore, if you're appending or inserting a blank record, SET RELATION positions the record pointer of the related file to the first record with a blank key.

When editing records in a view, dBASE commands such as EDIT, BROWSE, @...GET, and READ don't update related key fields when the *parent* key fields are changed. As a result, changing key fields is not recommended when a relation is set.

The SET FIELDS TO command does not check to make sure that files are related. You must make sure of this yourself.

The commands INDEX, LOCATE, SET FILTER, and SET RELATION ignore the SET FIELDS TO fields list entirely, regardless of the SET FIELDS ON/OFF status. Therefore, these commands *can* access fields in the active database file structure that are *not* in the fields list. To access fields in files from an unselected work area, you must specify the file alias with each of these fields.

The normal and recommended way to use INDEX, LOCATE, and SET RELATION is with fields in the active database file structure. However, SET FILTER commonly requires the use of fields in unselected work areas. Since a separate filter condition is maintained for each work area, take care when closing a database file. Filters which specify fields of database files closed in other work areas should also be removed.

This same precaution about closing files also applies in applications where INDEX, LOCATE, or SET RELATION is used with fields other than those belonging to the active database file. dBASE III PLUS does not check these conditions automatically because to do so would slow down processing time.

**NOTE**
To insure data integrity when using view files, or the SET FIELDS and SET RELATION commands, you should add records to one database file at a time, with all fields of the database file accessible.

Changes to key information should be done knowing that if you change *parent* key information without also changing corresponding key information in related files, you will lose the relation between those records.

Do not construct field lists from multiple unrelated files unless you have a sound reason for doing so.

**Accessibility**

Using multiple related files can bring up questions of ambiguity and the accessibility of certain fields. The rules governing accessibility are:

- Files unaffected by a SET FIELDS TO command retain their default fields lists.
- Fields from files unaffected by a SET FIELDS TO command are accessible if an alias is specified provided you opened the file (with USE) after issuing the SET FIELDS TO command.
- Fields of the active database file are accessible only if they are included in the fields list.
- Fields in related files are accessible if they appear on the fields list and an alias is specified.
- Within files that do have fields on the fields list, files not on the fields list are not accessible.
- Unless a file has related descendants, fields beyond its own listed fields are not displayed.

When referring to fields without the use of an alias, dBASE resolves ambiguities on these guidelines:

- The active database file is searched first.
- If the field is not in the active database file, the fields list is searched.
- If a field isn't found on the fields list for the active database file, then fields list entries for related files are searched.

## Examples

A simple use of SET FIELDS TO would be to limit the fields displayed in a single file.

```
. USE Travel
. SET FIELDS TO Firstname,Lastname,Cost
. LIST NEXT 5
Record#  FIRSTNAME        LASTNAME        COST
      1  Claire           Buckman      1199.00
      2  Rick             Lisbonn      1378.00
      3  Hank             Bicksby      3195.00
      4  Lena             Garnett      1969.00
      5  Lisa             Kafmanan      988.00
```

Using multiple files requires that the files be related.

```
. SELECT 1
. USE Travel
. SELECT 2
. USE Clients INDEX Cnames
. SELECT 1
. SET RELATION TO Lastname INTO B
. SET FIELDS TO Lastname,Travelcode,B->City,B->State
. LIST NEXT 7
Record#  LASTNAME        TRAVELCODE   CITY         STATE
      1  Buckman         CI10         Oxnard       CA
      2  Lisbonn         AV10         Glendale     CA
      3  Bicksby         KS19         Flagstaff    AZ
      4  Garnett         GE23         Reno         NV
      5  Kafmanan        HI14         Reseda       CA
      6  Johnson         KS19         Riverside    CA
      7  Collins         HI14         Portland     OR
```

## See Also

CLEAR FIELDS, CREATE/MODIFY VIEW, CREATE VIEW FROM ENVIRONMENT, SET VIEW TO

## SET FILTER

SET FILTER allows display of only those records of a database file that meet a specified condition.

### Syntax

SET FILTER TO [FILE <filename> /?] [<condition>]

### Defaults

SET FILTER TO ↵ without a condition turns off the filter for the active database file. SET FILTER TO adds a filter (query) file to a catalog if one is open and SET CATALOG is ON.

### Usage

SET FILTER applies only to the database file open in the work area where the command is issued. Therefore, a different filter condition may be set for each open database file.

All commands that require a database file to be in USE, such as AVERAGE, BROWSE, EDIT, and REPORT, can use conditions specified by SET FILTER.

SET FILTER TO FILE reads the filter condition from a query (.qry) file created by CREATE/MODIFY QUERY. If a catalog is open, you can use the catalog query clause (?) to display a menu of all query files for the active database file.

SET FILTER TO <condition> sets up a filter based on a valid dBASE expression. The condition can filter records in the active database file based on any of the allowed data types, for example, on a character field, `SET FILTER TO Lastname = 'Jones'`; or on a date field, `SET FILTER TO Departure = CTOD('01/01/86')`.

Filters aren't activated until *after* the record pointer is moved within a file. This means that your filter condition isn't in effect unless something has caused the pointer to change position since the condition was set. The best way to assure that a filter is in effect is to execute a GOTO TOP command after setting the filter.

## Examples

You can set a filter for the Clients database file to limit the records to those in which the state is NV (Nevada):

```
. USE Clients
. SET FILTER TO State = 'NV'
. GO TOP
. LIST Lastname, State
Record#  Lastname                State
      4  Garnett                 NV
     31  Rizzo                   NV
     43  Gilbert                 NV

. SET FILTER TO
. GO TOP
. LIST NEXT 5 Lastname, State
Record#  Lastname                 State
      1  Buckman                 CA
      2  Lisbonn                 CA
      3  Bicksby                 AZ
      4  Garnett                 NV
      5  Kafmanan                CA

. USE
```

## See Also

CREATE/MODIFY QUERY, SET DELETED

## SET FIXED

SET FIXED determines whether a fixed number of decimal places are displayed for all numeric output.

### Syntax

SET FIXED ON/OFF

### Defaults

SET FIXED is normally OFF.

If SET DECIMALS has not been used, the number of decimal places is two.

### Usage

When SET FIXED is OFF, the number of decimal places that are displayed in calculation results is exactly as described in the SET DECIMALS command.

When SET FIXED is ON, all numeric output displays exactly the number of decimal places designated by the SET DECIMALS command.

### See Also

SET DECIMALS

# SET FORMAT



## SET FORMAT

SET FORMAT allows the user to select custom forms stored as format (.fmt) files.

## Syntax

SET FORMAT TO [ < filename > /?]

## Defaults

If you do not specify a file extension with a format filename, the .fmt extension is assumed.

Each work area with an open database file can have an open format file. Both SET FORMAT TO ↵ and CLOSE FORMAT close the open format file in the currently selected work area.

The SET FORMAT TO command updates a catalog if one is open, and SET CATALOG is ON.

## Usage

If SET FORMAT TO is not used, APPEND, CHANGE, EDIT, and INSERT use the standard display and entry form dBASE III PLUS provides.

If a catalog is open, SET FORMAT TO ? displays a menu of all format files for the active database file.

Remember that you can use CREATE/MODIFY SCREEN, a menu-driven *screen painter,* to create format files. CREATE/MODIFY SCREEN first creates a screen (.scr) file, and then when you save that file, it creates a format (.fmt) file. You can use the Screen Painter to create a format file which you can use with SET FORMAT TO to APPEND or EDIT records in the associated database file.

When you want to make changes to the format file, you can make them with MODIFY SCREEN, through which both the .scr and .fmt files are updated. However, if you make changes to the format file using the MODIFY COMMAND text editor, the changes will not be made to the screen file and future use of MODIFY SCREEN will not reflect changes to the format file.

## Programming Notes

If you want to use a multi-page format (.fmt) file in which the @...SAY...GETs continue on from 2 to 32 screen pages, include a READ wherever you want a page break. The **PgDn** and **PgUp** keys flip the pages. Multi-page format files only work when the format file is opened with SET FORMAT TO.

## Example

To use the custom screen format file, Clients.fmt, with the active Clients database file:

```
. SET STATUS ON
. USE Clients
. SET FORMAT TO Clients
. EDIT
```

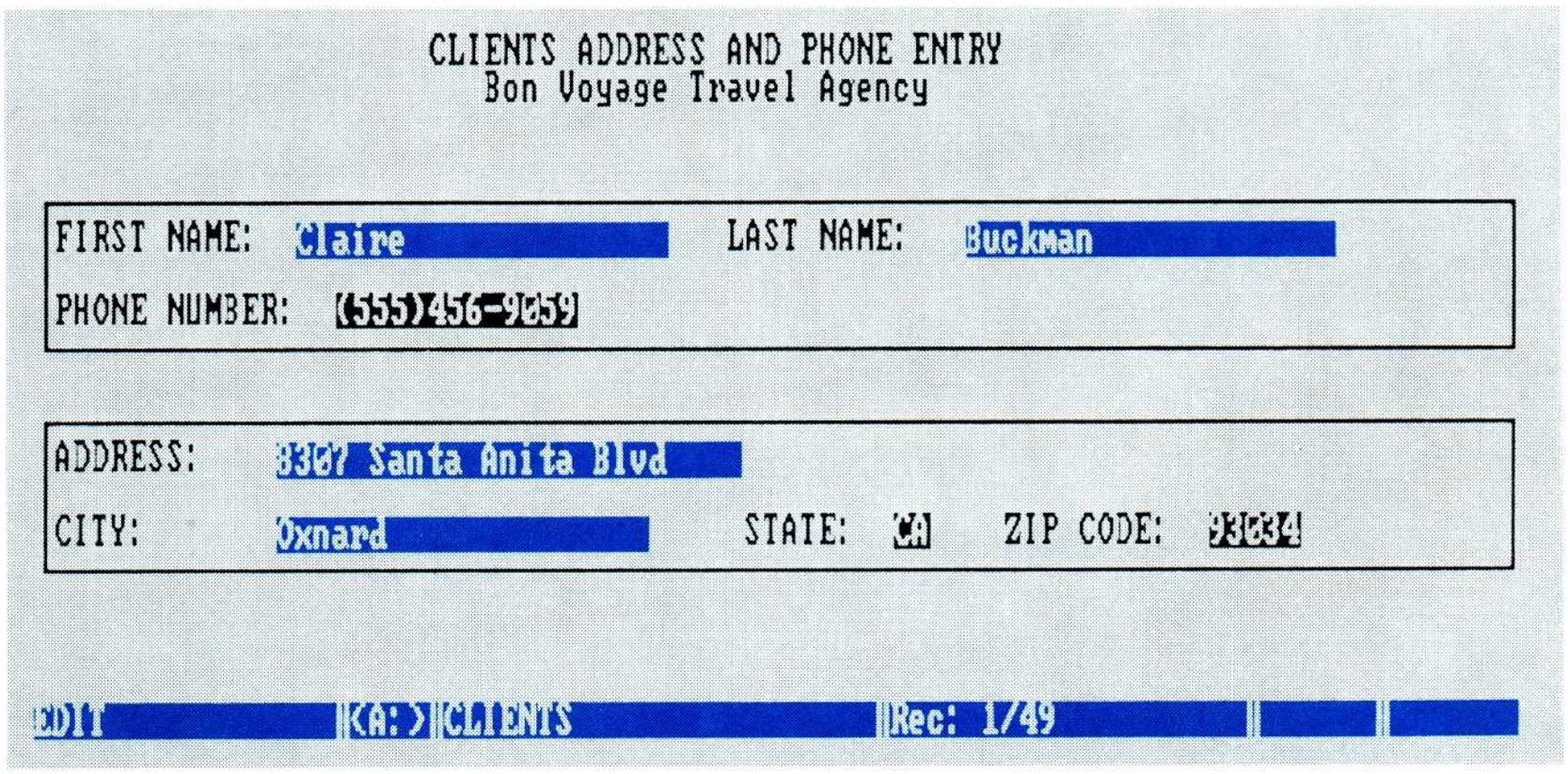



Figure 5-16 The EDIT command with a custom format file

## See Also

@, APPEND, CHANGE, CLOSE, EDIT, INSERT, READ, SET DEVICE

## SET FUNCTION

SET FUNCTION reprograms a function key. You may specify a character string expression of up to 238 characters for each function key. The sequence of characters you specify will then be transmitted to the computer each time you then press that function key.

### Syntax

SET FUNCTION < exp > TO < expC >

### Usage

The standard function key assignments are:

| Key | Value | Key | Value |
|---|---|---|---|
| **F1** | help; | **F6** | display status; |
| **F2** | assist; | **F7** | display memory; |
| **F3** | list; | **F8** | display; |
| **F4** | dir; | **F9** | append; |
| **F5** | display structure; | **F10** | edit; |

Function key **F1** cannot be reprogrammed.

The expression designating a function key can be numeric or character. You can use either type of expression if your terminal has numbered function keys. If not, you can use a character expression to enter the name of the programmable function keys.

A semicolon included in the string of characters specified for a function key assignment generates a carriage return.

### Examples

To assign the SET command to function key **F10**, so that pressing **F10** executes the command:

```
. SET FUNCTION 10 TO 'SET;'
```

To assign multiple commands to function key **F9**, so that the sequence of commands CLEAR, USE Clients, and LIST STRUCTURE is executed:

```
. SET FUNCTION 9 TO 'CLEAR;USE Clients;LIST STRUCTURE;'
```

## SET HEADING

SET HEADING determines whether or not column titles are shown above each field for DISPLAY, LIST, SUM, and AVERAGE.

### Syntax

SET HEADING ON/OFF

### Defaults

SET HEADING is normally ON.

### Usage

The commands DISPLAY, LIST, SUM, and AVERAGE display a column title for each displayed field, memory variable, or expression.

The column width is the length of the heading or field width, whichever is larger.

### Examples

This example uses the original version of the Tours database file on the Sample Programs and Utilities disk. (If you have altered this file, your results may be slightly different.)

```
. USE Tours
. DISPLAY Departure, Seats_open
Record#  Departure  Seats_open
      1  08/05/85           25

. SET HEADING OFF
. DISPLAY Departure, Seats_open
      1    08/05/85    25
```

## SET HELP

SET HELP determines whether the query **Do you want some help? (Y/N)** appears when a command is incorrectly entered at the dot prompt.

### Syntax

SET HELP ON/OFF

### Defaults

SET HELP is normally ON.

### Usage

When you make an error entering a command, and then respond with "Yes" to the HELP query, dBASE III PLUS displays the HELP screen for the command you entered. If you entered the command name incorrectly, dBASE III PLUS displays the main HELP menu.

### See Also

HELP

## SET HISTORY

SET HISTORY ON/off determines whether previously executed commands are stored in a *history* buffer. SET HISTORY TO specifies the desired number of executed commands to be stored in the *history* buffer.

### Syntax

SET HISTORY ON/OFF

SET HISTORY TO <expN>

### Defaults

SET HISTORY is normally ON. The default number of stored commands is 20. The allowable range of commands is an integer between zero and 16,000.

### Usage

SET HISTORY allows you to redisplay, edit, or re-execute commands (including those issued from The Assistant). You use ↑ and ↓ to redisplay commands you previously executed from the dot prompt. Commands are displayed from most recent to least recent as you step through the history buffer using the ↑ key. (Refer to Chapter 2 for more information on the operation of the history buffer.)

The maximum number of commands the history buffer stores is determined by SET HISTORY TO. When this number is reached, the earliest commands executed are cleared from the buffer. If you SET HISTORY to a number that is less than the number of commands currently stored in the history buffer, all stored commands are erased. Otherwise, they are retained.

Commands stored in the history buffer reside in memory. You can LIST HISTORY or DISPLAY HISTORY to view the entire contents of the history buffer at one time. The command SET HISTORY OFF clears the history buffer.

You can also store commands executed from dBASE programs if you SET DOHISTORY ON. You can redisplay, edit, and re-execute commands from dBASE programs. Edits you make to commands will not be saved in the program file, however.

The maximum size of the history buffer is limited by the amount of memory you have available, and by the memory occupied by other programs already resident in memory, including dBASE III PLUS. To calculate the amount of memory that the history buffer uses, add nine bytes to the number of bytes in each command.

**See Also** DISPLAY HISTORY, LIST HISTORY, SET DOHISTORY

## SET INDEX

SET INDEX opens the specified index files.

### Syntax

SET INDEX TO [ < list of index filenames > /?]

### Defaults

The file extension .ndx is assumed unless otherwise specified.

SET INDEX TO ↵ closes all open index files in the current work area. It is an alternative syntax for CLOSE INDEX.

### Usage

If a catalog is open, use SET INDEX TO ? to display a list of all index files for the active database file if a catalog is open.

Up to seven index files may be opened with SET INDEX if they all have been created from the database file in USE. Note that SET INDEX doesn't create the index files; it only opens them.

The first index file in the list is the master index, which controls the order of the database file. The master index file also specifies the index key expression for which you can specify a search value with FIND and SEEK. You can switch the index file designated as the master index with the SET ORDER command. You specify a number from 1 to 7, which corresponds to the order of the open index files in the file list of the currently selected work area.

Not just the master but all open index files are automatically updated whenever a change is made to the associated database file. You can restore the natural order of the database file with SET ORDER TO 0; however, if you make any changes to the index keys or add records, you'll need to reindex the file.

If you APPEND records to a database file which uses an index file, new records are appended to the end of the database file. After you complete field entries in an appended record, the record will be repositioned according to its position determined by the index key expression of the master index file. The INSERT command is equivalent to APPEND when an index file is in use.

### Record Pointer

When you issue SET INDEX TO, the database record pointer is positioned at the beginning of the file as determined by the master index file.

## Example

To open the Clients database file and select the Location and Cnames index files, with Location as the master index file, you would type:

```
. USE Clients
. SET INDEX TO Location, Cnames
```

## See Also

CLOSE, INDEX, REINDEX, SET ORDER, USE

# SET INTENSITY

## SET INTENSITY

SET INTENSITY determines whether the enhanced screen attribute is used for full-screen operations such as APPEND and EDIT.

## Syntax

SET INTENSITY ON/OFF

## Defaults

SET INTENSITY is normally ON.

## Usage

When SET INTENSITY is ON, dBASE III PLUS displays @...SAY commands with the *standard* screen attribute and displays @...GET commands using the *enhanced* screen attribute.

Normally, the standard attribute provides white letters on a black background, and the enhanced attribute specifies black letters on a white background. The settings for standard and enhanced attributes are made with the SET COLOR command.

If SET INTENSITY is OFF, the enhanced screen attribute is not used. Instead, the standard attribute is used for both @...GET and @...SAY commands.

## See Also

SET COLOR

# SET MARGIN



## SET MARGIN

SET MARGIN adjusts the left-hand margin for all printed output. The video display is unaffected.

## Syntax

SET MARGIN TO <expN>

## Defaults

The left-hand margin defaults to a value of zero.

## Usage

SET MARGIN TO sets the margin for all printed output from commands which can direct output to the printer, for example, DISPLAY and LIST, as well as the ? command (with SET PRINT ON) and @...SAY commands (with SET DEVICE TO PRINT).

The SET MARGIN setting is added to the left margin settings specified with CREATE/MODIFY REPORT or LABEL.

## Example

To set the left margin 10 spaces to the right of the first possible print position:

```
. SET MARGIN TO 10
```

## SET MEMOWIDTH

SET MEMOWIDTH TO adjusts the width of memo field output.

### Syntax

SET MEMOWIDTH TO <expN>

### Defaults

The default memo width is 50; the minimum is 8 characters.

### Usage

Use SET MEMOWIDTH TO to alter the column width of memo fields during output.

## SET MENU

SET MENU determines whether a menu of cursor movement keys appears with the full-screen commands.

### Syntax

SET MENU ON/OFF

### Defaults

SET MENU is normally ON.

### Usage

In full-screen mode, the **F1** key toggles the menu display on and off.

## SET MESSAGE TO

SET MESSAGE TO displays a user-defined character string on the bottom line of the screen provided SET STATUS is ON.

### Syntax

SET MESSAGE TO [ < expC > ]

### Defaults

SET MESSAGE TO ↵ resets the message to a null string, in which case only dBASE III PLUS messages are displayed. The message does not display unless SET STATUS is ON.

### Usage

The character string can have a maximum of 79 characters. If you specify a character string, it must be delimited with 'single quotes', "double quotes", or [brackets]. You can also specify a character type memory variable or display a field which contains the literal text string.

In full-screen menu programs, such as The Assistant, you can't set your own message; the application messages override them.

The message line is in effect when you are entering commands from the dot prompt and with full-screen editing commands such as APPEND, EDIT, and READ.

### Tips

If you are using the status bar and message line, it is a good idea not to specify lines 22 and 24 in your format file. Otherwise, the display will be confusing. That way, with SET STATUS ON, you can use SET MESSAGE to convey useful information on Line 24.

### See Also

SET STATUS

## SET ODOMETER

The SET ODOMETER command defines the update interval of the record counter for commands which display a record count.

### Syntax

SET ODOMETER TO <expN>

SET ODOMETER determines the interval at which the record counter is updated on the screen for commands such as COPY and RECALL. The default interval is one.

The record counter can be removed from the screen entirely using SET TALK OFF.

### Example

Issue the following commands with SET TALK ON and notice the difference in the record count that is displayed on the screen:

```
. USE Travel
. SET ODOMETER TO 10
. COPY TO Temp
```

## SET ORDER

SET ORDER sets up any open index file as the master (controlling) index, or removes control from all open index files, without closing and reopening the files.

### Syntax

SET ORDER TO [ < expN > ]

> **NOTE**
> After a SET ORDER command, you must position the record pointer in a database file with a GOTO, FIND, or SEEK record-positioning command. This activates the index file so that records appear in the indexed order.

### Usage

The numeric expression must be an integer between 0 and 7, depending on the number of open index files. The number you specify corresponds to the position of a file in list of index files specified by SET INDEX TO or with the USE command (that is, Index1, Index2, and so on). If you specify 0, records in the database file appear in their natural, or unindexed, order, and no index files are updated.

An index file must be open to be updated. If you add or change records in your database file without an index file open, you'll have to reindex any associated index files to include the new or changed data.

### Tip

SET ORDER saves time when you want to reassign a controlling index file, because it does not close and open the index files or move the record pointer.

## Examples

The following examples use the Clients database file with the Location index file, indexed on State; the Cnames index file, indexed on Lastname; and the Zipcode index file, indexed on Zipcode.

```
. USE Clients INDEX Location,Cnames,Zipcode
LIST OFF NEXT 9 Lastname,State,Zipcode
Lastname             State Zipcode
Bicksby              AZ    86001
Goreman              AZ    85201
Youngblood           AZ    85344
Dickerson            AZ    85041
Campbell             AZ    72450
Hamilton             AZ    85749
Long                 AZ    85745
Buckman              CA    93034
Lisbonn              CA    91206
```

To change the master index to the third index file specified with the USE command, you would type the following:

```
. SET ORDER TO 3
Master index: B:Zipcode.ndx
. GOTO TOP
. LIST OFF NEXT 9 Lastname, State, Zipcode
Lastname             State Zipcode
Campbell             AZ    72450
Dickerson            AZ    85041
Goreman              AZ    85201
Youngblood           AZ    85344
Long                 AZ    85745
Hamilton             AZ    85749
Bicksby              AZ    86001
Adams                NM    87501
Gilbert              NV    89106

. USE
```

## See Also

NDX()

## SET PATH

SET PATH TO specifies the route that dBASE III PLUS follows to find files that are not found in the current directory.

### Syntax

SET PATH TO [ <path list> ]

### Defaults

SET PATH TO ↵ without additional parameters releases any dBASE search path set previously with SET PATH TO. By default, dBASE III PLUS will only search the current directory.

### Usage

SET PATH enables you to access your files in a tree-structured directory system. It specifies the sequence of directories that dBASE III PLUS can search to find specified files that are not found in the current directory.

The current disk directory is the one from which you started dBASE III PLUS, unless it was changed using the DOS CHDIR (CD) command or you changed drives with the SET DEFAULT command. Only the current directory is searched, unless you use the SET PATH command to specify an alternate search path.
dBASE III PLUS does not use the paths established by the DOS PATH command, nor does DOS use the path that dBASE III PLUS establishes with SET PATH.

A *path* is a list of directories separated by backslashes and listed in order from the top level directory to the level where the file is located, for example A:\DBASE\DATA.

A *path list* is a series of paths separated by commas or semicolons, up to 60 characters in length, which specifies more than one directory search path.

### Tips

SET PATH does not affect the DIR command listing. The DIR command lists files in the current or specified directory, not in the directory set through a path.

SET PATH applies only to commands that search for existing files. If you create a file and the file is to be placed in a directory other than the current directory, you must explicitly specify the path name with the filename.

## Example

The following SET PATH command specifies two additional directories to be searched if a file (in this case, Program.prg) is not found in the current default drive directory. The next search directory is the \Baseline directory on drive A. If the Program.prg file is not found there, dBASE III PLUS will search for the file in the \Dbase\Newpgms directory on drive B.

```
. SET PATH TO A:\BASELINE,B:\DBASE\NEWPGMS
. DO Program
```

## See Also

DIR, SET DEFAULT

## SET PRINT

SET PRINT ON directs all output not formatted with the @...SAY command to the printer as well as to the screen.

### Syntax

SET PRINT ON/OFF

### Defaults

SET PRINT is normally OFF.

### Usage

When SET PRINT is ON, all screen output, including TALK, LIST, and ?, is routed to the printer.

### Special Cases

Use SET DEVICE TO PRINT to direct @...SAY commands to the printer. SET PRINT ON does not affect @...SAY commands.

### See Also

SET DEVICE

## SET PRINTER TO

SET PRINTER TO selects a DOS device for printer output.

### Syntax

SET PRINTER TO < DOS device >

### Defaults

The default DOS printer device is LPT1. dBASE directs all printer output to this device unless told otherwise with SET PRINTER.

### Usage

SET PRINTER names the DOS device to which printer output is sent. The devices are the parallel printer ports LPT1, LPT2, LPT3, and the serial communication ports COM1 and COM2.

If your computer has available ports, it may be connected to more than one printer or more than one serial device. There are both parallel and serial types of printers. Check your printer manual for your printer type.

### Examples

Suppose you have a LaserJet printer that requires serial communication (connected to serial port:COM1), and you need to print the Travel report for the Travel database file. Assuming that you already have the baud rate set, you would type the following:

```
. SET PRINTER TO COM1              && Select serial port 1
. USE Travel INDEX Trips
. REPORT FORM Traveler TO PRINT
```

You would need to set the baud rate with the DOS MODE command before you start dBASE III PLUS and run this example. Typically, to set up the laser printer, you would type the following command from the drive and directory containing the MODE program:

```
C > MODE COM1:9600,N,8,1,P
```

### See Also

SET PRINT

## SET PROCEDURE

SET PROCEDURE opens a named procedure file.

### Syntax

SET PROCEDURE TO [ < procedure filename > ]

### Defaults

The procedure filename must include the drive location if the named file is not on the default drive or in a dBASE search path.

If no file extension is specified for the procedure filename, dBASE III PLUS assumes the .prg extension.

SET PROCEDURE < filename > opens the specified procedure file. SET PROCEDURE TO ↵ or CLOSE PROCEDURE closes the current procedure file.

### Usage

A procedure file may contain a maximum of 32 procedures (routines). The command PROCEDURE marks the beginning of each new routine. Only one procedure file may be open at a time.

### Example

The following program and procedure files illustrate the use of procedures. For example, you could create the following program which opens a procedure file and executes the three procedures within the open procedure file:

```
* Setproc.prg - demonstrates SET PROCEDURE TO
SET PROCEDURE TO Proctest    && Open the procedure file
Do Proc1                     && Execute procedure 1
Do Proc2                     && Execute procedure 2
CLOSE PROCEDURE              && Close the procedure file
* EOF: Setproc.prg
```

The setup of the Proctest procedure file (which is also a dBASE program file) could be as simple as the following:

```
* Proctest.prg - Procedures
PROCEDURE Proc1
   ? "This is a message from Proc1."
RETURN
PROCEDURE Proc2
   ? "Greetings from Proc2."
    DO Proc3
RETURN
PROCEDURE Proc3
   ? "Called from Proc2."
RETURN
* EOF: Proctest.prg
```

To invoke the procedure (if both the dBASE program file and the procedure file are on the default disk drive), you would type:

```
. DO Setproc
This is a message from Proc1.
Greetings from Proc2.
Called from Proc2.
```

## See Also

DO, PARAMETERS

## SET RELATION

SET RELATION links two open database files according to a key expression that is common to both files.

### Syntax

SET RELATION TO [ < key expression > / < expN > INTO < alias > ]

### Defaults

SET RELATION TO ↵, without any additional parameters, removes the relation from the currently selected work area.

### Usage

SET RELATION links the active database file to an open database file in another work area. The INTO file is identified by its alias. Only one relation can be set from each work area. The active database file that controls the relation is referred to as the *parent* file. The file which is linked to the parent file by a relating expression is referred to as the *child*.

### Options

With the key expression option, the key must be contained in the parent database file, and the child file must be indexed on that key. Whenever the parent file is repositioned, the child file is searched for the first record matching the key expression from the active file.

With the numeric expression option, the active database is always linked to the record number in the file specified by the numeric expression. The linked file must not be indexed. Typically, the numeric expression will be the RECNO() function. This causes record 1 in the parent file to be linked to record 1 in the child, record 2 in the parent to record 2 in the child, and so on.

### Record Pointer

If a matching record cannot be found in the linked file, the linked file is positioned at the end of the file (a blank record), and EOF( ) is True.

### Tips

You can save a working environment, including relations, with CREATE VIEW < .vue filename > FROM ENVIRONMENT. Remember that CREATE/MODIFY VIEW is available to you as a menu-driven command for linking files. It also allows you to save relations to a .vue file.

## Examples

Following the steps in this example, you can use data from two database files simultaneously.

```
. SELECT 1
. USE Travel
. SELECT 2
. USE Clients INDEX Cnames
. Select 1
. SET RELATION TO Lastname INTO B
. LIST NEXT 7 Lastname, B->Zipcode
Record#  Lastname              B->Zipcode
      1  Buckman               93034
      2  Lisbonn               91206
      3  Bicksby               86001
      4  Garnett               89504
      5  Kafmanan              91355
      6  Johnson               92504
      7  Collins               97219
```

In this example, the Travel database file is opened in work area one. The Clients database file is opened in work area two, indexed on Lastname plus Firstname by the Cnames index file. The Lastname field is used as the link between the two files. Lastname is also a part of the index key expression of Cnames. From work area one, the example lists the Lastname field from the parent file, Travel, and the related Zipcode field in the child, the Clients file, referenced in work area two by alias B.

To relate the two files used in the previous example but change the relations to a numeric expression (by using RECNO()), you could enter the following:

```
. SELECT B
. SET INDEX TO
. SELECT A
. SET RELATION TO RECNO( ) INTO B
```

## See Also

CREATE/MODIFY VIEW, SET FIELDS, SET VIEW

## SET SAFETY

SET SAFETY provides a level of protection against overwriting or otherwise destroying files.

### Syntax

SET SAFETY ON/OFF

### Defaults

SET SAFETY is normally ON.

### Usage

With SET SAFETY ON, if a file with the same name as the one you're creating already exists, dBASE III PLUS prompts
< filename > **already exists, overwrite it? (Y/N)**

If you want to overwrite the existing file, you would respond Yes. The existing file is then overwritten and the data it contains is lost. If you don't want to overwrite the existing file, you would respond No. You could then use this opportunity to rename the newly created file.

SET SAFETY OFF means you could overwrite files or destroy data without receiving a warning.

### See Also

COPY, COPY FILE, CREATE < newfile > /MODIFY STRUCTURE, INDEX, JOIN, SAVE, SORT, TOTAL, UPDATE, ZAP

## SET SCOREBOARD

SET SCOREBOARD determines whether dBASE III PLUS messages appear on line 0 if SET STATUS is OFF.

### Syntax

SET SCOREBOARD ON/OFF

### Defaults

SET SCOREBOARD is normally ON.

### Usage

With SET SCOREBOARD ON and SET STATUS OFF, you see messages on line 0 of the screen. These include **Del** to indicate that a record is marked for deletion in full-screen editing, and the correct range when you enter a value that is outside a specified RANGE.

With SET SCOREBOARD OFF and SET STATUS OFF, dBASE III PLUS does not display these indicators.

### Special Cases

The SET SCOREBOARD command has no effect when SET STATUS is ON.

### See Also

SET STATUS

## SET STATUS

SET STATUS determines whether the status bar displays at the bottom of the screen when you are working from the dot prompt, and in full-screen editing commands such as APPEND, EDIT, and READ.

### Syntax

SET STATUS ON/OFF

### Defaults

STATUS is normally set ON by the Config.db file loaded during installation of single-user dBASE III PLUS. If the Config.db file is not used, SET STATUS defaults to OFF.

### Usage

When SET STATUS is ON, the status bar displays the following information about the current working environment:

Figure 5-17 Status bar

Note that the scoreboard and record number are on the status bar. When SET STATUS is OFF, the scoreboard information is at the top right of the screen unless SET SCOREBOARD is also OFF.

After you issue commands that cause information on the status bar to change, the status bar is updated.

SET STATUS OFF has no effect when you are in a full-screen application, such as ASSIST or CREATE/MODIFY REPORT. There, it remains ON at all times. However, after leaving these commands and returning to the dot prompt, STATUS returns to its OFF setting.

### Tips

SET STATUS ON can be used with SET FORMAT and READ. However, be sure you do not specify fields or text below line 21 so you do not overwrite the status bar or message lines below it.

### Programming Notes

If SET STATUS is ON, the status bar displays when you APPEND or EDIT from within a command file. This makes the screen resemble a dBASE III PLUS application command, such as ASSIST.

### See Also

SET MESSAGE TO, SET SCOREBOARD

## SET STEP

SET STEP halts the execution of a program after each instruction. It allows you to step through a program one line at a time. The command is primarily a debugging tool.

### Syntax

SET STEP ON/OFF

### Defaults

SET STEP is normally OFF.

### Usage

During operations with SET STEP ON, a single program instruction is executed at a time. The result of each operation is displayed. Then, before the next instruction is executed, the following message appears:

**Press SPACE to step, S to suspend,or Esc to cancel...**

Program processing pauses until you enter one of the choices.

### See Also

SET DEBUG, SET ECHO, SET TALK

## SET TALK

SET TALK determines whether the response to certain dBASE III PLUS commands is displayed.

### Syntax

SET TALK ON/OFF

### Defaults

SET TALK is normally ON.

### Usage

SET TALK is usually ON during interactive use. It displays the following information as each record or command is processed: record numbers, memory variables, and the results of commands such as APPEND FROM, COPY, PACK, STORE, and SUM.

SET TALK OFF is used in programs to control what appears on the screen and on the printer.

### Example

To see the difference with SET TALK ON and OFF, store information to memory variables.

```
. STORE 'Start' TO mvar1
Start
. SET TALK OFF
. STORE 'Finish' to mvar2
. SET TALK ON
```

Note that after you SET TALK OFF, a command response is not displayed.

### See Also

SET DEBUG, SET ECHO

## SET TITLE

SET TITLE turns the catalog file title prompt on and off.

### Syntax

SET TITLE ON/OFF

### Defaults

SET TITLE is normally ON.

### Usage

Ordinarily, you are prompted for a catalog file title when adding a new file to a catalog. With SET TITLE OFF, this prompt does not appear. To add a title to a file's catalog record later, you must use commands such as EDIT or REPLACE to edit the title field in the catalog file title open in work area 10.

### See Also

EDIT, REPLACE, SET CATALOG

# SET TYPEAHEAD TO

## SET TYPEAHEAD TO

SET TYPEAHEAD TO specifies the size of the type-ahead buffer.

## Syntax

SET TYPEAHEAD TO <expN>

## Defaults

The default number of characters that the type-ahead buffer holds is 20. The allowable range is an integer between 0 (zero) and 32,000.

## Usage

SET TYPEAHEAD only works when SET ESCAPE is ON.

To completely disable the type-ahead buffer, SET TYPEAHEAD TO 0. This deactivates both the ON KEY command and the INKEY() function.

In full-screen EDIT or APPEND operations, the type-ahead buffer will only hold 20 characters regardless of a previous setting with SET TYPEAHEAD TO.

## Tips

If you execute an error-handling program using ON ERROR DO <program>, it is a good idea to include SET TYPEAHEAD TO 0 as the first line of the error-handling program. Because an error is an unanticipated condition, disabling the type-ahead buffer is a safety precaution.

## Special Cases

If you try to enter more than the specified number of characters into the type-ahead buffer, all of the extra characters are lost. If SET BELL is ON, the bell will sound. If this happens repeatedly, you should increase the size of the type-ahead buffer.

## See Also

INKEY(), ON, SET BELL, SET ESCAPE

## SET UNIQUE

SET UNIQUE determines whether all records with the same key value are included in the index (.ndx) file.

### Syntax

SET UNIQUE ON/OFF

### Defaults

SET UNIQUE is normally OFF.

### Usage

If you create a new index file while SET UNIQUE is ON, and several records have the same key value, only the first record that dBASE III PLUS encounters with that value is included in the new index file.

SET UNIQUE is the same as issuing INDEX ON <key expression> TO <filename> UNIQUE. When several records have the same key value, only the first record that dBASE III PLUS encounters with that value is included in the index file.

Whenever you REINDEX an index file that was created with UNIQUE, the file retains its UNIQUE status, regardless of whether SET UNIQUE is ON or OFF.

Adding new records or editing existing records in the database file with a unique index engaged retains the UNIQUE status of the index file. This means that if you APPEND a record with an index key that is already in the index file, that record does not become a part of the index file, although it gets added to the database file. Similarly, EDITing a record and changing its key to one which is already in the index file removes that record from the index file, even though the record is still in the database file.

### Examples

This example shows how you can get an alphabetical list of all the states in which your clients live. The index file consists of only the first record that contains each state.

```
. USE Clients
. SET UNIQUE ON
. INDEX ON State TO State
100% indexed              5 records indexed
```

> **NOTE**
> You could also have created the unique index file by typing:
> INDEX ON State TO State UNIQUE.

To display the five unique records, each containing a different state, you would type:

```
. LIST State, Lastname
Record#  State   Lastname
      3  AZ      Bicksby
      1  CA      Buckman
     15  NM      Adams
      4  NV      Garnett
      7  OR      Collins

. SET UNIQUE OFF
. REINDEX
100% indexed           5 Records indexed
```

The index file retains its UNIQUE status. Just to make sure, you could LIST the records again:

```
. LIST State, Lastname
Record#  State   Lastname
      3  AZ      Bicksby
      1  CA      Buckman
     15  NM      Adams
      4  NV      Garnett
      7  OR      Collins
```

## Programming Note

To restore an index file to a non-unique status, rebuild the index with SET UNIQUE OFF and the INDEX ON command.

## See Also

FIND, INDEX, REINDEX, SET INDEX, USE

## SET VIEW TO

SET VIEW TO opens a view (.vue) file, setting up the database files and a SET RELATION previously established using CREATE/MODIFY VIEW or CREATE VIEW FROM ENVIRONMENT.

### Syntax

SET VIEW TO < .vue filename > /?

### Defaults

SET VIEW TO updates a catalog if one is open and SET CATALOG is ON.

> **NOTE**
> If database files included in a view are not already in a catalog when you issue the SET VIEW TO command, they will not be added to the open catalog.

SET VIEW TO opens the database files and SETs a RELATION previously established in the view file. You never need to close a view file directly. To close the files within a view, use the CLEAR ALL or CLOSE command.

### Usage

Either enter a view filename as part of the syntax or use the catalog query ? to open a menu of all view (.vue) files in the open catalog.

A view consists of:

- All open database files, index files, and the work area number of each
- All relations between the database files
- The currently selected work area number
- The active field list
- The active filter
- The open format (.fmt) file, if any

**Tips**

To change the contents of the view (.vue) file, use MODIFY VIEW.

**See Also**

CREATE VIEW FROM ENVIRONMENT, CREATE/MODIFY VIEW, SET FIELDS, SET FILTER, SET FORMAT, SET RELATION

## SKIP

SKIP moves the record pointer forward or backward in the active database file.

## Syntax

SKIP [ < expN > ]

## Record Pointer

If an index file is in use with the database file, SKIP follows the order of the index. If the file in use is not indexed, the record pointer moves sequentially by record number.

The record pointer moves forward unless the result of the numeric expression is negative. If the expression is not included, dBASE III PLUS moves forward one record.

If SKIP is issued when the record pointer is on the last record in a file, the value of RECNO( ) is one greater than the last record number in the file, and EOF( ) is True.

If SKIP -1 is issued when the record pointer is on the first record, RECNO( ) remains at 1, but BOF( ) (beginning-of-file) is True.

## Examples

The database file Tours on the Sample Programs and Utilities disk is used for the following examples.

To move forward one record in the database:

```
. USE Tours
. SKIP
Record No.     2
```

To move forward five records:

```
. STORE 5 TO mskip
5
. SKIP mskip
Record No.     7
```

To move backward three records:

```
. SKIP -3
Record No.     4
```

## See Also

BOF( ), EOF( ), RECNO( )

## SORT

SORT creates a new database file in which the records of the active database file are reordered alphabetically, chronologically, or numerically by the specified key fields.

## Syntax

SORT < scope > TO < newfile > ON < field > [/A] [/C] [/D] [, < field2 > [/A] [/C] [/D]...] [WHILE < condition > ] [FOR < condition > ]

## Defaults

The TO file has the .dbf extension unless otherwise specified.

SORTs are in ascending order (/A) unless otherwise specified. Ascending order is the same as ASCII order (refer to Appendix D).

SORT updates a catalog if one is in use.

## Usage

You may not SORT logical or memo fields.

SORT does not work with substrings or other expressions. Use INDEX to use these as a key and then COPY to create a sorted file.

You can SORT on a maximum of ten fields.

A file cannot be SORTed to itself or to any other open file.

## Options

The keywords ASCENDING and DESCENDING are alternatives to specifying /A for ascending order and /D for descending order.

/C causes the SORT not to differentiate between upper and lower case.

## Tips

Do not use the letters A through J or M as database filenames, because they are reserved for default ALIAS names. However, AA is a valid database filename.

You may combine /C with either /A or /D. When using two options, only include one slash (for example, /DC).

When SORTing on multiple fields, the most important key is placed first. Separate field names with commas.

## Examples

The Travel database is to be SORTed on the initials of the selling agent. The field name is Agent:

```
. USE Travel
. SORT ON Agent TO Agent
  100% Sorted        49 Records sorted... Copying text file.
```

You can SORT the same Travel database file to compare sales among agents. The fields involved are Agent and Cost. The following example creates a SORTed file in order by Agent, then in descending numeric order by Cost.

```
. SORT ON Agent,Cost/D TO Csales
  100% Sorted        49 Records sorted... Copying text file
```

## See Also

INDEX

## STORE

STORE creates and initializes one or more memory variables.

### Syntax

STORE < expression > TO < memory variable list >

An alternative syntax for STORE is:

< memory variable > = < expression >

If a memory variable of the same name already exists, it is overwritten.

### Usage

A single expression may be stored to several memory variables with a single STORE command. With the alternative syntax, < memvar > = < expression >, you may only store an expression to one memory variable at a time.

If you create a memory variable using the syntax < memvar > = < expression >, the memory variable name cannot be the same as a dBASE III PLUS command.

A numeric variable may be added to itself; a character variable may be concatenated to itself.

The maximum number of active memory variables is 256. A maximum of 6,000 bytes is allocated to memory variables unless changed in the Config.db file.

### Special Cases

A field with the same name as a memory variable takes precedence in all operations except with macro substitution (&). To explicitly specify that the memory variable has precedence, use the following representation with **m->**, the symbol used to specify a memory variable:

m-> < memvar >

### Examples

To store the logical False condition (.F.) to the memory variable Truth:

```
. STORE .F. TO truth
.F.
```

To store the number 0 to the memory variables A, B, and C:

```
. STORE 0 TO a,b,c
0
```

To raise the commission by 15% and store the new value to the memory variable Newcom:

```
. USE Tours
. Newcom = Percent * 1.15
12.07500
```

To create and use a memory variable that has the same name as a field in the active database:

```
. USE Clients
. STORE 'XX' TO state
XX
. ? state
CA
```

To view the memory variable:

```
. ? m->state
XX
```

## SUM

SUM totals expressions involving numeric fields in the active database file.

## Syntax

SUM [ < scope > ] [ < expression list > ] [TO < memvar list > ] [WHILE < condition > ] [FOR < condition > ]

## Defaults

Unless otherwise specified by a scope or a WHILE or FOR clause, all database records are SUMmed.

If no expression list is present, all numeric fields are SUMmed.

## Examples

The Tours database on the Sample Programs and Utilities disk is used for the following example. To SUM the Unitcost and Seats_open fields and store the results in the memory variables Tcost and Tseats:

```
. USE Tours
. SUM Unitcost, Seats_open TO tcost, tseats
    30 records summed
    Unitcost     Seats_open
    46263.00     1314
. ? tcost, tseats
    46263.00     1314
```

## See Also

AVERAGE

## SUSPEND

SUSPEND is primarily a debugging tool that lets you stop a program while it is executing. You can then enter commands from the keyboard or the history buffer, and RESUME the program.

## Syntax

SUSPEND

## Usage

If you press the **Esc** key while a program is running, dBASE III PLUS prompts **Cancel, Ignore, or Suspend? (C, I, or S)**.

If you respond with **S**, you can either change a command line stored in the history buffer or execute new commands. You cannot modify the SUSPENDed dBASE program file or other open dBASE program or procedure files. If you try, dBASE III PLUS responds with **File is Already Open**.

When you are ready to continue the execution of the program, type RESUME.

You can also SUSPEND execution by placing the SUSPEND command within a program.

If you create any memory variables while the file is SUSPENDed, they are PRIVATE at the level of the suspension.

CANCEL cancels all DO files, including ones that are SUSPENDed. You must close any procedure file with CLOSE PROCEDURE.

## See Also

CANCEL, RESUME, RETURN, SET ECHO, SET STEP

## TEXT

TEXT is used in programs to output blocks of text to the screen or printer. The command provides a simple, convenient way to write to the output device.

## Syntax

TEXT
< text characters >
ENDTEXT

## Usage

The text characters are output exactly as originally entered into a program.

The & (Macro) function is not expanded when embedded in the text, and is output exactly as typed.

## Example

An example of the TEXT command is shown below:

```
TEXT
This is an example of text that is to be displayed by the
TEXT command. This text is routed directly to the screen
without being interpreted by dBASE III PLUS.

ENDTEXT
```

## TOTAL

TOTAL sums the numeric fields of the active database file TO a second database file. The numeric fields in the TO database file contain the totals for all records that have the same key value in the original database.

## Syntax

TOTAL ON <key field> TO <filename> [<scope>] [FIELDS <fields list>] [WHILE <condition>] [FOR <condition>]

## Defaults

The TO filename must include the drive designator if it is not on the default drive. A .dbf extension will be assumed unless otherwise specified.

Unless otherwise specified by the scope, or the WHILE or FOR clause, all records are TOTALed. All numeric fields are totalled unless limited by the FIELDS list.

If a catalog is in use, the TO file is added to it.

## Usage

The active database file must be either INDEXED or SORTed on the key field. If the TO filename exists, TOTAL does not overwrite the existing file without a warning prompt, unless SET SAFETY is turned OFF.

All records with the same value in the specified key field become a single record in the TO database. All numeric fields appearing in the FIELDS list contain totals. All other fields contain the data from the first record in the key field set.

The structure of the TO file is identical to that of the active database file, except that memo fields are not copied to the new file.

## Tips

Do not use the single letters A through J or M as database filenames because they are reserved for ALIAS names. However, AA is a valid database filename.

If a field size in the TO file is not large enough to accommodate the total, dBASE III PLUS places asterisks in the field indicating an overflow. To avoid this, use MODIFY STRUCTURE to increase the size of the fields in the active database file to a size that can handle the largest total.

## Examples

```
. USE Travel
. INDEX ON Agent TO Agent
  100% indexed          49 Records indexed
. TOTAL ON Agent TO Agent
     49 Record(s) totalled
      4 Records generated
. USE Agent
. LIST Agent, Cost
Record#  Agent      Cost
      1  JT     24863.00
      2  LB      9650.00
      3  MM     19705.00
      4  RA     21808.00
. USE
```

## TYPE

TYPE displays the contents of a text file containing only ASCII-value characters.

## Syntax

TYPE <filename> [TO PRINT]

## Defaults

The filename must include the file extension. If the file is not on the default drive, and a SET PATH is not set to the file's location, you must precede the filename with its drive and directory location.

Files with extended ASCII characters will be displayed as graphic characters.

## Usage

TYPE will display standard ASCII files only. dBASE III PLUS database files, index files, and database text files cannot be read. TYPE cannot specify an open file. In other words, a program may not include a command to TYPE itself or any other open file.

## UPDATE

UPDATE uses data from an existing file to replace fields in the active database file. The changes are made by matching records in the two database files based on a single key field.

### Syntax

UPDATE ON <key field> FROM <Alias> REPLACE <field> WITH <exp> [,<field2> WITH <exp> ...] [RANDOM]

### Usage

The database being UPDATEd must be the active database file; the FROM database file is an open database file in another work area, specified by its alias.

The key field must have the same field name in both database files. If the key field in the UPDATEd file is not unique for each database record, only the first record that matches the key field value is updated.

### Options

Both files must be either sorted or indexed on the key field unless RANDOM is used. RANDOM requires the UPDATEd database to be indexed on the key field, however, the FROM file may be in any order.

If the REPLACE expression involves a field in the FROM database (which is usually the case), the WITH field must be identified as alias->field.

## Example

The following example shows how the number of seats available on trips in the Tours database file can be UPDATEd to reflect the seats taken by vacationers in the Travel database file. The seats remaining are those that can still be sold.

```
. SELECT 2
. USE Travel
. SELECT 1
. USE Tours
. INDEX ON Travelcode TO Tcodes
  100% indexed                30 Records indexed
. UPDATE ON Travelcode FROM Travel REPLACE Seats_open WITH Seats_open-1 RANDOM
       49 records updated
. LIST Travelcode, Seats_open
Record# Travelcode Seats_open
     16 A026                17
     22 AS15                43
      1 AV10                24
     15 CE16                23
      28 CH22               33
     10 CI10                73
      5 CN7                 85
     14 CO14                61
        .
        .
        .
     25 WA8                 36
. CLOSE DATABASES
```

The Seats_open field now reflects the new number of seats available (for instance, A026 originally had 19 seats available, and now has 17).

## USE

USE opens an existing database file and up to seven index files in the selected work area. If the database contains memo fields, the associated .dbt file is opened automatically.

## Syntax

USE [ < filename > /?] [INDEX < index file list > ] [ALIAS < alias name > ]

## Defaults

The USE command without any additional parameters closes the active database and any associated index files in the currently selected work area.

Unless otherwise specified, dBASE III PLUS assumes the .dbf extension for the filename and the .ndx extension for the files specified in an index file list.

If the filename isn't already recorded, USE < filename > updates a catalog if one is open and SET CATALOG is ON.

## Usage

USE ? displays a menu of cataloged .dbf files when a catalog is open. The index file list can consist of up to seven files, all associated with the database file in USE.

If no alias name is included, the name given to the alias is the same as the database filename. You can also use the letters A through J to select work areas 1 through 10. When you select a new work area, the database file in that work area becomes the active database file.

## Record Pointer

When you USE a database file without any index files, the record pointer is positioned at the first record in the file.

When you USE a database file with one or more index files, the record pointer is positioned at the first logical record of the first index file listed. The record pointer follows the order of that index file.

## Examples

To open the Clients database file in the currently selected work area without index files and without specifying an alias, you would type the following:

```
. USE Clients
```

To open this database with two index files and set the alias to CUSTOMERS:

```
. SELECT 1
. USE Clients INDEX Location,Cnames ALIAS Customers
```

Then, if you've changed work areas, you can select work area 1 again to make the Clients database file the active database file by typing:

```
. SELECT Customers
```

## See Also

CLOSE, INDEX, SELECT, SET INDEX, SET VIEW

## WAIT

WAIT causes all dBASE III PLUS processing to pause until any key is pressed.

## Syntax

WAIT [ < prompt > ] [TO < memvar > ]

## Defaults

If a ↵ or other nonprintable character is entered, a null character (ASCII value 0) is entered in the memory variable.

## Usage

The prompt may be a character variable or a literal character string. If no prompt is specified, **Press any key to continue...** displays.

## Options

If the TO < memvar > option is included, a character type memory variable is created which stores the keyboard entry.

## Special Cases

If you use ON KEY WAIT TO < memvar > , the key that was pressed which caused the program to branch to the ON KEY command is stored in the memory variable.

## Examples

You can use WAIT to halt execution of a dBASE program temporarily. To do this, you could include the following commands in your dBASE program:

```
WAIT 'Do you wish to continue? (Y/N) ' TO Mcon
IF UPPER(Mcon) = 'Y'
   RETURN
ENDIF
```

The $ substring operator evaluates the key pressed in response to the WAIT command to see if it is the upper-case letter A.

## See Also

ON KEY

## ZAP

ZAP removes all records from the active database file.

## Syntax

ZAP

## Usage

ZAP is equivalent, but faster, to a DELETE ALL command followed by PACK.

If SET SAFETY is ON, dBASE III PLUS responds with the prompt **ZAP** <filename>**? (Y/N).**

Any open index files in the current work area are automatically reindexed to reflect the empty database file.

dBASE III PLUS reclaims the space previously used by the ZAPped file contents, including that of any associated memo or index files.

## See Also

DELETE, PACK, SET SAFETY

# Chapter 6
# FUNCTIONS

## Using Functions

Functions perform specialized operations that augment and enhance the dBASE III PLUS commands. They are used to evaluate or convert data and return a result.

dBASE III PLUS functions all have () — opening and closing parentheses — after the function name (except for the & macro function). The () may or may not contain data to be evaluated.

This chapter contains an alphabetic listing of the functions available in dBASE III PLUS. Each listing contains the syntax and usage of the function, plus examples when necessary.

> **TIP**
> The working examples of dBASE III PLUS functions provided in this chapter use files from the Sample Programs and Utilities disk. All examples assume that you're working with an untouched copy of the original file.

## Input and Output Data Types

With the exception of the & (macro substitution) function, you specify a function by entering its name followed immediately by the input to the function (enclosed in parentheses). The parentheses are necessary whether or not the function requires input, so that dBASE III PLUS can recognize your entry as a function. Functions always return a value and must be used as expressions or in expressions following a command verb. The functions and the data types they produce are listed alphabetically by function in the following table:

| Function Name | Description | Input Data Type | Output Data Type |
|---|---|---|---|
| & | Macro substitution | C | C,D,L,N |
| ABS | Absolute value | N | N |
| ASC | Character to ASCII code conversion | C | N |
| AT | Substring search | C | C |
| BOF | Beginning-of-file | * | L |
| CDOW | Character day of week | D | C |
| CHR | ASCII code to character conversion | N | C |
| CMONTH | Calendar month | D | C |
| COL | Current screen column position | * | N |
| CTOD | Character to date conversion | C | D |
| DATE | System date | * | D |
| DAY | Day of month | D | N |
| DBF | Name of database file in USE | * | C |
| DELETED | Deleted record | * | L |
| DISKSPACE | Free space (bytes) on disk | * | N |
| DOW | Day of week | D | N |
| DTOC | Date to character conversion | D | C |
| EOF | End-of-file | * | L |
| ERROR | Number for ON ERROR condition | * | N |
| EXP | Exponential $e^x$ | N | N |

An * indicates that this function does not require any input.

## FUNCTIONS

| Function Name | Description | Input Data Type | Output Data Type |
|---|---|---|---|
| FIELD | Names of fields in database file | N | C |
| FILE | File existence | C | L |
| FKLABEL | Names of function keys | N | C |
| FKMAX | Maximum number of function keys | * | N |
| FOUND | Result of database file search | * | L |
| GETENV | Operating system environmental variables | C | C |
| IIF | Immediate IF | C,D,N | C,D,N |
| INKEY | Keypress during program execution | * | N |
| INT | Integer | N | N |
| ISALPHA | Evaluate for letter | C | L |
| ISCOLOR | Evaluate for color mode | * | L |
| ISLOWER | Evaluate for lower case | C | L |
| ISUPPER | Evaluate for upper case | C | L |
| LEFT | Substring selection from left side | C | C |
| LEN | Length of character string | C | N |
| LOG | Logarithm | N | N |
| LOWER | Upper to lower case conversion | C | C |
| LTRIM | Remove leading blanks | C | C |
| LUPDATE | Last update of database file | * | D |
| MAX | Determine greater of two values | N | N |
| MESSAGE | ON ERROR message string | * | C |
| MIN | Determine smaller of two values | N | N |
| MOD | Modulus | N | N |
| MONTH | Month of year | D | N |
| NDX | Names of open index files | N | C |
| OS | Name of operating system | * | C |
| PCOL | Printer column position | * | N |
| PROW | Printer row position | * | N |
| READKEY | Determine full-screen exiting keypress | * | N |
| RECCOUNT | Number of records in database file | * | N |
| RECNO | Current record number | * | N |
| RECSIZE | Size of record | * | N |
| REPLICATE | Repeat character expression | C | C |

An * indicates that this function does not require any input.

| Function Name | Description | Input Data Type | Output Data Type |
|---|---|---|---|
| RIGHT | Substring selection from right side | C | C |
| ROUND | Rounds off | N | N |
| ROW | Current screen row position | * | N |
| RTRIM | Remove trailing blanks | C | C |
| SPACE | Generates blank spaces | N | C |
| SQRT | Square root | N | N |
| STR | Numeric to character conversion | N | C |
| STUFF | Replace portion of string | C | C |
| SUBSTR | Substring selection | C | C |
| TIME | System time | * | C |
| TRANSFORM | Character/number in PICTURE format | C,N | C |
| TRIM | Remove trailing blanks | C | C |
| TYPE | Validates expression | C | C |
| UPPER | Lower to upper case conversion | C | C |
| VAL | Character to numeric conversion | C | N |
| VERSION | dBASE III PLUS version number | * | C |
| YEAR | Year | D | N |

An * indicates that this function does not require any input.

## Classification

The functions are grouped by functional classification in the table below:

| Functional Classification | Function | |
|---|---|---|
| | Name | Description |
| Date | CDOW | Day of week |
| | CMONTH | Calendar month |
| | CTOD | Character to date conversion |
| | DATE | System date |
| | DAY | Day of month |
| | DOW | Number of day of the week |
| | DTOC | Date to character conversion |
| | MONTH | Month of year |
| | YEAR | Year |

| Functional Classification | Function | |
|---|---|---|
| | Name | Description |
| Character Manipulation | AT | Substring search |
| | LEFT | Substring selection from left side |
| | LOWER | Forces lower case |
| | LTRIM | Remove leading blanks |
| | REPLICATE | Repeat character expression |
| | RIGHT | Substring selection from right side |
| | RTRIM | Remove trailing blanks |
| | SPACE | Generates blank spaces |
| | STUFF | Replace portion of string |
| | SUBSTR | Substring selection |
| | TRANSFORM | Character/number in PICTURE format |
| | TRIM | Remove trailing blanks |
| | UPPER | Forces upper case |
| Mathematical | ABS | Absolute value |
| | EXP | Exponential ($e^x$) |
| | INT | Integer |
| | LOG | Logarithm |
| | MAX | Greater of two values |
| | MIN | Smaller of two values |
| | MOD | Modulus |
| | ROUND | Rounds off |
| | SQRT | Square root |
| Conversion | ASC | Character to ASCII code |
| | CHR | ASCII code to character |
| | CTOD | Character to date conversion |
| | DTOC | Date to character conversion |
| | LOWER | Upper to lower case |
| | STR | Numeric to character |
| | UPPER | Lower to upper case |
| | TRANSFORM | Numeric to character |
| | VAL | Character to numeric |

| Functional Classification | Function | |
|---|---|---|
| | **Name** | **Description** |
| Specialized Tests | BOF | Beginning-of-file |
| | COL | Current screen column position |
| | DELETED | Deleted record |
| | DISKSPACE | Free space (bytes) on disk |
| | EOF | End-of-file |
| | ERROR | Number for ON ERROR condition |
| | FILE | File existence |
| | FOUND | Result of database file search |
| | IIF | Immediate IF |
| | ISALPHA | Evaluate for letter |
| | ISCOLOR | Evaluate for color mode |
| | ISLOWER | Evaluate for lower case |
| | ISUPPER | Evaluate for upper case |
| | LEN | Length of character string |
| | LUPDATE | Last update of database file |
| | MESSAGE | ON ERROR message string |
| | PCOL | Printer column position |
| | PROW | Printer row position |
| | RECCOUNT | No. of records in database |
| | RECNO | Current record number |
| | RECSIZE | Size of record |
| | ROW | Current screen row position |
| | TIME | System time |
| | TYPE | Validates expression |
| Identification | DBF | Name of database file in USE |
| | FIELD | Names of FIELDS in database files |
| | FKLABEL | Names of function keys |
| | FKMAX | Maximum number of function keys |
| | GETENV | Operating system environment variables |
| | NDX | Names of open index files |
| | OS | Name of operating system |
| | VERSION | dBASE III PLUS version number |
| Input | INKEY | Keypress during program execution |
| | READKEY | Full-screen exiting keypress |

## &

& is the macro substitution function. It substitutes the contents of a memory variable for its name in those cases where dBASE III PLUS would otherwise take the variable name literally. This function can be used only for character variables.

## Syntax

& <character variable> [.<cstring>]

## Usage

The macro substitution function is used to obtain the contents of a character memory variable when dBASE III PLUS expects a literal value rather than a character expression. It can also be used to convert a character memory variable to a number. Some examples of when dBASE III PLUS expects a literal value are FIND, USE, and other commands that use a filename. Of course, the commands and their keywords are also literal values.

## Special Cases

Field names normally take precedence over memory variables when their names are the same. Therefore, the syntax *m->*<*memvar*> is used when a memory variable with the same name should take precedence over a field name. Since the & function always expects a memory variable as its argument, this syntax is unnecessary and redundant.

## Examples

To use the FIND command with a memory variable:

```
. STORE 'Loftus' TO mname
Loftus
. USE Clients INDEX Cnames
. FIND &mname
. DISPLAY Lastname
Record#  Lastname
     34  Loftus
```

dBASE III PLUS normally takes the character sequence following FIND literally. Without the &, dBASE III PLUS searches the database for the character sequence *mname* instead of *Loftus*. With the &, dBASE III PLUS searches for the contents of Mname.

To substitute the contents of a memory variable name in a character string:

```
. STORE 'Loftus' TO name
Loftus
. STORE 'Hello &name' TO greeting
Hello Loftus
. ? greeting
Hello Loftus
```

To insert the contents of a memory variable into a character sequence that contains no blanks:

```
. STORE 'Cash' TO Income1
Cash
. STORE 'Savings' TO Income2
Savings
. STORE 'Checking' TO Income3
Checking
. STORE 'Income' TO Account
Income
. ? &Account.2
Savings
```

When nonblank characters are to be appended to the contents of a memory variable, as in the third example, use a period to indicate the end of the memory variable name.

To USE a database the name of which is in a memory variable called Dname:

```
. dname = 'Clients'
Clients
. USE &dname
. ? DBF()
B:Clients.dbf
```

DBF( ) is a function which displays the name of the active database file.

Note that the following two examples are most appropriate in a command file.

You can use the & function to tell dBASE III PLUS which command to execute:

```
. ACCEPT "Enter a dBASE III PLUS command: " TO cmd
Enter dBASE III PLUS command: <command>
. &cmd
```

To position the record pointer, you can enter:

```
. USE Travel
. STORE "15" TO rec
15
. GOTO &rec
```

## ABS( )

The ABS( ) function returns a number representing the absolute value of a numeric expression.

## Syntax

ABS( < expN > )

## Usage

The ABS( ) function enables you to find the difference between two numbers without first needing to test for which is larger.

The response is always a positive number.

## Examples

To find the absolute value of a numeric expression:

```
. i = 15
15
. j = 60
60
. ? ABS(i - j)
         45
. ? ABS(j - i)
         45
```

## ASC()

The ASC() function evaluates the leftmost character of a character string and returns its value in the IBM extended character set. Note that the ASCII values (decimal 0 through 127) are a subset of this set (decimal 0 through 255).

## Syntax

ASC ( <expC> )

## Examples

```
. ? ASC('Nestle')
  78
. STORE '123' TO number
123
. ? ASC(number)
  49
```

## AT()

The AT() function returns a number that shows the starting position of a character string within a second string. The contained character string is called a substring. If the substring is not contained within the character string, the function returns a zero (0).

## Syntax

AT( < expC > , < expC > )

## Examples

```
. ? AT('is a', 'This is a cat')
         6
```

The substring "is a" begins with the sixth character of the character string "This is a cat".

```
. STORE 'This is a cat' TO alpha
This is a cat
. STORE 'his' TO beta
his
. ? AT(beta,alpha)
         2
```

Using the Clients database, check the Travelplan field for Cruise:

```
. USE Clients
. ? AT('Buckman', Lastname) <> 0
.T.
. SKIP
Record No.     2
. ? AT('Buckman', Lastname ) <> 0
.F.
```

## See Also

LEFT(), RIGHT(), SUBSTR()

## BOF()

BOF() indicates the beginning of the file. This function is intended for programming applications in which the database is read in reverse order. A logical True (.T.) is returned when an attempt is made to move the record pointer before the first logical record of the active file.

## Syntax

BOF()

## Special Cases

If no database is in USE, BOF() returns .F.

## Examples

To test for the beginning of the file:

```
. USE Clients
. ? BOF()
.F.
. SKIP -1
Record No.    1
. ? BOF()
.T.
```

This sample segment of a command file tests for the beginning of a file while searching the database in reverse order for a particular Agent:

```
SET TALK OFF
USE Travel
GO BOTTOM
DO WHILE .NOT. BOF()
   IF Agent = 'JT'
      ? RECNO()
   ENDIF
   SKIP -1
ENDDO
SET TALK ON
```

## See Also

EOF()

## CDOW()

The CDOW() function returns the name of the day of the week from a date expression.

## Syntax

CDOW( <expD> )

## Usage

The date expression is a memory variable, a field, or any function that returns date type data.

## Examples

If the system date is 05/15/84:

```
. ? CDOW(DATE())
Tuesday
```

To store the day of the week from the system date to the memory variable Dayofweek.

```
. STORE CDOW(DATE()) TO dayofweek
Tuesday
. ? dayofweek
Tuesday
```

To translate a date into the name of the day of the week for use in standard text:

```
. ? DATE()
05/15/84
. STORE DATE()+9 TO testdate
05/24/84
. ? 'We will visit you a week from '+CDOW(testdate)
We will visit you a week from Thursday
```

## See Also

DAY(), DOW()

## CHR()

The CHR() function converts a number to a character. It enables you to produce special effects on a printer and/or monitor by sending a value in the IBM Extended Character Set for any keyboard character or graphics character. Note that ASCII code values (decimal 0 through 127) are a subset of this set (decimal 0 through 255).

## Syntax

CHR( < expN > )

## Usage

< expN > must be an integer from 1 to 255.

CHR(0), which is a null character, cannot be sent to the printer or any device. However, some printers allow CHR(128) to be interpreted as CHR(0). If the printer supports 8-bit characters, use printer escape commands or DIP switches to set the eighth bit off before sending CHR(128).

In comparisons, CHR(0) *must* be on the left side of the equation. When dBASE III PLUS performs character string comparisons, it reads what is on the right side first. Because CHR(0) is a null string, if it is on the right side, dBASE III PLUS reads no further and returns a .T.

## Examples

To display the ASCII character with a code (decimal value) of 65:

```
. ? CHR(65)
A
```

On most systems, the bell is sounded by means of ASCII code 7. Note that there is no keyboard equivalent for CHR(7). To ring the bell:

```
. ? CHR(7)
```

To ring the bell and write descriptive text on the screen:

```
. ? CHR(7) + 'Be more careful'
Be more careful
```

# CHR()

CHR() provides the ability to display characters for which there is no keyboard equivalent. For example, CHR(24) displays an up arrow.

```
. ? CHR(24)
↑
```

To use CHR(0) in a comparison, CHR(0) must be on the left side of the equation. In the following example, two single quotes are used in lines 5 and 9.

```
. Mem = 'abc'
abc
. ? Mem = CHR(0)
.T.
. ? Mem = ""
.T.
. ? CHR(0) = Mem
.F.
. ? "" = Mem
.F.
```

## See Also

ASC(), INKEY()

## CMONTH()

The CMONTH() function returns the name of the month from a date expression.

## Syntax

CMONTH( < expD > )

## Usage

The date expression is a memory variable, a field, or any function that returns date type data.

## Examples

If the system date is 05/15/84:

```
. ? CMONTH(DATE())
May
```

To store the name of the month from the system date to the memory variable Mmonth:

```
. STORE CMONTH(DATE()) TO mmonth
May
. ? mmonth
May
```

To determine which month is 76 days from the current date:

```
. ? CMONTH(DATE()+76)
July

. ? 'We will visit you in '+ CMONTH(DATE()+76)
We will visit you in July
```

## See Also

MONTH()

## COL()

The COL() function returns the current column position of the cursor. COL() makes it easier to control cursor positioning from within a program.

## Syntax

COL()

## Usage

You may store the value of COL() to a memory variable.

COL() may be used for relative addressing. For instance, @ 1, COL() + 5 is the same as @ 1, $ + 5. Both position the cursor five columns to the right of the current position.

> **NOTE**
> $ is a special operator that can also be used with @...SAY...GET commands to indicate the current row or column position on the display, or the coordinates on a printed page.

## Examples

Use COL() within a program to find the current cursor position on the screen. For example, if you want to end a loop when the cursor is at column 70, part of the program might include:

```
DO WHILE COL() < 70
   @ 5,COL() + 5 SAY 'X'
ENDDO
```

To print the sentence "This is relative addressing" starting on row 7, column 10:

```
@ 7, 10 SAY 'This is '
@ 7, COL() SAY 'relative addressing'
```

## See Also

@, PCOL(), PROW(), ROW()

## CTOD()

The CTOD() function converts a date that has been entered or stored as a character string to a date type variable. The format of the character string is normally mm/dd/yy, but this format can be changed by SET DATE and SET CENTURY.

## Syntax

CTOD( < expC > )

## Defaults

A twentieth-century date is assumed if you only use two numbers for the year.

## Usage

The character string used by CTOD() can range from "01/01/100" to "12/31/9999". There are no date type literal strings. All comparisons and assignments require use of the CTOD() function to convert literal strings into a date type variable.

CTOD() is the only way to create a date variable from a literal value.

## Examples

You could use CTOD to initialize a date memory variable converting a blank date string (" / / ") or a string specifying an exact date (for example, "01/01/80"). You could then use the date variable, for example, with an @...GET command, to enter a new date in the memory variable as it's displayed on your screen.

```
. STORE CTOD('01/01/80') TO testdate
01/01/80
. ? testdate
01/01/80
```

To convert a character variable to a date variable:

```
. STORE '01/01/80' TO testdate
01/01/80
. ? TYPE('testdate')
C
. STORE CTOD(testdate) TO newdate
01/01/80
. ? TYPE('newdate')
D
```

## See Also

SET CENTURY, SET DATE, DTOC()

## DATE( )

The DATE( ) function returns the system date in the form mm/dd/yy unless the format has been changed by SET CENTURY or SET DATE.

### Syntax

DATE( )

### Usage

The system date must be set at the DOS level.

### Examples

To display the system date:

```
. ? DATE()
05/04/84
```

To store the system date to the memory variable Mdate:

```
. STORE DATE() TO mdate
05/04/84
```

or

```
. mdate = DATE()
05/04/84
```

### See Also

SET CENTURY, SET DATE

## DAY()

DAY() returns the numeric value of the day of the month from a date expression.

## Syntax

DAY( <expD> )

## Usage

The date expression is a memory variable, a field, or any function that returns date type data.

## Examples

If the system date is 05/15/84:

```
. ? DAY(DATE())
15
```

To store the day from the system date to the memory variable Mday:

```
. STORE DAY(DATE()) TO mday
15
. ? mday
        15
. ? TYPE('mday')
N
```

## See Also

CDOW(), DOW()

## DBF()

The DBF( ) function returns the name of the database file in USE in the currently selected work area.

## Syntax

DBF( )

## Defaults

If no database file is open in the currently selected work area, DBF( ) returns a null string.

## Usage

This function is useful in programming to enable the program to work with a file that was in use before the program began.

## Examples

To determine which file is in USE when you are in the interactive mode:

```
. USE Clients
. ? DBF()
B:Clients.dbf
. CLOSE DATABASES
. ? DBF()

.
```

Within a program, to determine if a database file is in USE:

```
USE Travel
Null = ''                          && Use two single quotes
IF Null < DBF()
   * If true, save the file name.
   Old_file = DBF()
   *
   * Open a new file, and work with it.
   USE Clients
   DISPLAY STRUCTURE
   *
   * Reopen the original files.
   USE &Old_file                   && Note the & function
   DISPLAY STRUCTURE
ENDIF
USE
```

## See Also

FIELD( ), LUPDATE( ), NDX( ), RECCOUNT( ), RECSIZE( )

## DELETED( )

The DELETED( ) function identifies records that are marked for deletion. A logical True (.T.) is returned if the current record is marked for deletion; if not, a logical False (.F.) is returned.

## Syntax

DELETED( )

## Special Cases

If no database file is in USE, DELETED( ) returns a .F.

## Examples

To display all records in the active database file that are marked for deletion:

```
. DISPLAY ALL FOR DELETED()
```

To determine if the current record is marked for deletion:

```
. USE Clients
. ? DELETED()
.F.
. DELETE
      1 record deleted
. ? DELETED()
.T.
. RECALL
      1 record recalled
.? DELETED ()
.F.
```

## See Also

SET DELETED ON

## DISKSPACE()

The DISKSPACE() function returns an integer representing the number of bytes available on the default drive.

### Syntax

DISKSPACE()

### Programming Notes

Use DISKSPACE() with RECCOUNT() and RECSIZE() in application programs that automatically back up database files. This ensures there is sufficient space on the disk for the backup file. See the example in the RECSIZE() entry.

### Examples

The following program segment checks to see if there is sufficient disk space to create a new SORTed file from a file already in USE. A sort file requires the same amount of disk space as the original database file. This example also allows a safety factor of 4,000 bytes.

```
SET TALK OFF
USE Tours
STORE (RECCOUNT() * RECSIZE()) + 2000 TO mfilesize
IF DISKSPACE() > mfilesize*2
   SORT ON Lastname TO Newfile
ELSE
   @ 10,16 SAY "There is not enough space to sort the file."
ENDIF
SET TALK ON
```

### See Also

GETENV(), OS(), RECCOUNT(), RECSIZE(), VERSION()

## DOW()

The DOW() function returns a number that represents the day of the week from a date expression, with Sunday being day 1.

## Syntax

DOW( < expD > )

## Usage

The date expression is a memory variable, a field, or any function that returns date type data.

## Examples

If the system date is 05/15/84:

```
. ? DOW(DATE())
3
```

To store the day of the week from the system date to the memory variable Dayofweek:

```
. STORE DOW(DATE()) TO dayofweek
3
. ? dayofweek
          3
```

To determine the day of the week code for a date 197 days in the future:

```
. ? DOW(DATE() + 197)
4
```

## See Also

CDOW(), DAY()

## DTOC()

### Syntax

DTOC( < expD > )

### Examples

The DTOC( ) function converts a date to a character string.

To store the system date as a character type memory variable:

```
. STORE DTOC(DATE()) TO testdate
05/15/84
. ? TYPE('testdate')
C
```

To connect a date variable to a character string for use as text:

```
. STORE DATE() TO testdate
05/15/84
. ? TYPE('testdate')
D
. STORE "The date is '+DTOC(testdate) TO text
Today's Date is: 05/15/84
```

### See Also

CTOD( ), SET CENTURY

## EOF()

EOF() indicates the end of a file. A logical True (.T.) is returned when the last logical record of the active database file is passed.

## Syntax

EOF()

## Usage

When EOF() is True, RECNO() is RECCOUNT() + 1. Also, certain commands (such as SKIP) return the error message **End of File encountered.**

## Special Cases

If no database file is in USE, EOF() returns .F.

## Examples

To test for the end of a file:

```
. USE Clients
. GO BOTTOM
. ? EOF()
.F.
. SKIP
Record no. 50
. ? EOF()
.T.
```

## See Also

BOF(), ERROR(), FOUND()

## ERROR( )

The ERROR( ) function returns the number corresponding to the error that caused an ON ERROR condition.

### Syntax

ERROR( )

### Usage

ERROR( ) traps errors so corrective action may be taken if recovery is possible. Corrective actions may include closing files, changing disks, deleting files to make space, or redisplaying menus to choose other options.

Some errors, of course, are not recoverable. Normal termination is done with the CANCEL command. Termination in debugging mode is done with the SUSPEND command. See Chapter 7 for the number that is returned by ERROR( ) for a particular error message.

> **NOTE**
> For this function to return a number, an ON ERROR command must be active. RETURN and RETRY clear the error number and message that dBASE III PLUS recorded.

### Examples

Here is a sample error procedure that illustrates how to use the ON ERROR command to recover from the error indicated by the message **File is already open**:

```
*MAIN.PRG
ON ERROR DO Err_prog WITH ERROR()
*
*
MODIFY COMMAND Screen.fmt     &&If Screen.fmt is in use, close it.
*
*
*EOF:MAIN.PRG
```

The Err_prog procedure looks like this:

```
*Err_prog.prg
PARAMETERS Error_no
IF Error_no = 3
   CLOSE FORMAT
   RETRY
ENDIF
```

## See Also

MESSAGE(), ON ERROR, RETRY

## EXP( )

The EXP( ) function returns the value of $e^x$.

## Syntax

EXP( < expN > )

## Usage

The numeric expression is the value of the exponent, x, in the equation $e^x$.

## Examples

To determine the value of $e^{1.000}$:

```
. ? EXP(1.000)
     2.718

. STORE 1.000 TO y
1.000
. ? EXP(y)
     2.718
```

## See Also

SET DECIMALS

## FIELD( )

The FIELD( ) function returns the field name in the specified numeric position in the file structure of the currently selected database file.

## Syntax

FIELD( < expN > )

## Usage

dBASE III PLUS returns a null string if the numeric expression is not between 1 and 128, or if the numeric expression refers to an unused field (for example, if you have a file structure with 28 field names, and you ask for the 29th field name).

Because the FIELD( ) function addresses each field by its number, it enables you to handle fields as an array.

The FIELD( ) function returns all field names in upper case letters.

## Examples

Using the Clients database file on the Sample Programs and Utilities disk, the following example demonstrates this function:

```
. USE Clients
. ? FIELD(3)
ADDRESS
. ? FIELD(4) = 'CITY'
.T.
. ? FIELD(4) = 'City'
.F.
. number = 2
2
. ? FIELD(number)
LASTNAME
. fld = FIELD(number + 2)
CITY
```

To write a program that determines the number of fields in a file:

```
SET TALK OFF
USE Clients
num_fields = 0
null = ""
DO WHILE null < FIELD(num_fields + 1)
   num_fields = num_fields + 1
ENDDO
? num_fields
SET TALK ON
```

## See Also

DBF(), LUPDATE(), NDX(), RECCOUNT(), RECSIZE()

## FILE( )

The FILE( ) function verifies the existence of a specified filename and returns logical True (.T.) if the file exists.

## Syntax

FILE( < expC > )

## Usage

The FILE( ) function requires that the character expression input include both the filename and the file extension. If the file resides on a different drive or directory than the default, a path must be set using the SET PATH TO command or the drive and directory must be specified as part of the filename. FILE( ) is not case-sensitive to the letters in a filename; any combination of upper and lower case letters is accepted.

## Examples

If the default drive is A and Clients.dbf is on drive B:

```
. ? FILE('Clients.dbf')
.F.
. ? FILE('B:Clients')
.F.
. ? FILE('B:Clients.dbf')
.T.
. STORE 'B:Clients.dbf' TO test
B:Clients.dbf
. ? FILE(test)
.T.
```

If Names.dbf is in the C:\TEMP directory and the logged directory is C:\DBASE:

```
. SET PATH TO
. ? FILE("NAMES.DBF")
.F.
. SET PATH TO C:\TEMP\
. ? FILE("NAMES.DBF")
.T.
```

## FKLABEL( )

The FKLABEL( ) function returns a character containing the name of the function key corresponding to the specified numeric expression.

## Syntax

FKLABEL( < expN > )

## Usage

Computer keyboards identify the function keys in different ways. For example, some keyboards use numbers such as 1 through 10, while others use descriptive labels such as SETUP.

The FKLABEL( ) function allows you to use SET FUNCTION to redefine programmable function keys that have non-numeric labels.

Any numeric expression between one and FKMAX( ) is valid.

> **NOTE**
> FKLABEL(1) corresponds to the second function key on keyboards that do not have a separate Help key.

## Examples

To determine the number of programmable function keys on a computer, so you can set a function for each key accordingly, include the following lines in a program:

```
i = 1
DO WHILE i <= FKMAX()      && Until count = last key
   ACCEPT "SET " + FKLABEL(i) + " TO " TO string
   SET FUNCTION FKLABEL(i) TO string
   i = i+1
ENDDO
*
* Display the new settings...
LIST STATUS
```

## See Also

FKMAX( ), SET FUNCTION

## FKMAX()

The FKMAX() function returns an integer representing the number of programmable function keys on the computer keyboard.

## Syntax

FKMAX()

## Usage

The FKMAX() function allows you to set the function keys to suit the computer keyboard you're using.

## Examples

To determine the number of programmable function keys on a computer so you can set a function for each key accordingly, include the following lines in a program:

```
i = 1
DO WHILE i <= FKMAX()
   ACCEPT "SET " + FKLABEL(i) + " TO " TO string
   SET FUNCTION FKLABEL(i) TO string
   i = i+1
ENDDO
*
* Display the new settings...
*
LIST STATUS
```

## See Also

FKLABEL(), SET FUNCTION

## FOUND()

The FOUND() function returns a logical True (.T.) if the previous FIND, SEEK, LOCATE, or CONTINUE command was successful.

## Syntax

FOUND()

## NOTES

FOUND() is usually used in programming to determine which branch to take, depending upon whether or not a search command finds the requested data. There is one FOUND() per work area.

If files are linked by SET RELATION, dBASE III PLUS sets FOUND() to True (.T.) or False (.F.) in the related work area when the record pointer moves in the current database file. When you move the record pointer in the current database file, dBASE III PLUS does an implicit search in any RELATEd database files.

If you move the record pointer with any command other than FIND, LOCATE, SEEK, or CONTINUE, the result of FOUND() is .F.

## Examples

The following examples use the Travel database.

To check the results of a search on an unINDEXed file, a program might contain:

```
LOCATE FOR Agent ='JT'
DO WHILE FOUND()
   ? Agent
   CONTINUE
ENDDO
```

To check the results of a search on a file INDEXed on Agent, a program might contain:

```
SEEK 'JT'
IF FOUND()
   DO WHILE Agent = 'JT'
      ? Agent
      SKIP
   ENDDO
ENDIF
```

## See Also

CONTINUE, EOF(), FIND, LOCATE, SEEK, SET RELATION

## GETENV( )

The GETENV( ) function returns the contents of an environmental system variable such as PATH or COMSPEC. The name of the system variable must be entered as a character string.

## Syntax

GETENV( < expC > )

## Usage

GETENV( ) enables you to check the environmental variables of the operating system. The name of the system variable must be specified in a character expression.

If the character expression is not found, dBASE III PLUS returns a null string.

See your DOS manual for more information on the SET command and environmental variables.

## Examples

To find the contents of any DOS SET parameters you've set up (in this example, a SET PATH):

```
. ? GETENV("COMSPEC")
C:\bin\command.com
. ? GETENV("PATH")
\bin;D:\;A:\
```

## See Also

DISKSPACE( ), OS( ), VERSION( )

## IIF( )

IIF( ) stands for "Immediate IF" and is a shortcut to the IF...ENDIF construction. It works both interactively from the dot prompt and within programs.

## Syntax

IIF( < expL > , < exp1 > , < exp2 > )

## Usage

If the logical expression is True, IIF( ) returns the result of the first expression, otherwise it returns the result of the second expression. < exp1 > and < exp2 > can be character, numeric, or date expressions. However, both must be the same type. The result is the same data type as the two expressions.

You can use the IIF( ) function from the dot prompt, or with commands such as CREATE/MODIFY REPORT and CREATE/MODIFY LABEL. The IIF( ) function allows you to replace a field or a value in an expression with one of two values, based on the evaluation of a condition.

## Tips

Replacing IF...ENDIF structures in programs with IIF( ) improves execution time.

## Examples

In the following program, you can replace the first set of commands with the one IIF( ) command line that immediately follows them. This example uses the IF...ENDIF structure:

```
IF Sex = 'F'
   mname = 'Ms. ' + Last_name
ELSE
   mname = 'Mr. ' + Last_name
ENDIF
```

This is the one-line replacement using IIF( ):

```
mname = IIF(Sex = 'F', 'Ms. ', 'Mr. ') + Last_name
```

You could use this IIF( ) construction to print labels with the appropriate salutation.

If you have a database file with the date fields Start_date and End_date, you can extend the End_date by 30 days if that date has already passed.

In the interactive mode:

```
. REPLACE ALL End_date WITH IIF(End_date > DATE(), End_date, End_date + 30)
```

If you have already assigned a value to the logical memory variable Is_error, the following routine converts the variable to a number, then saves it in the numeric memory variable Error_code. If Is_error is True, the new memory variable is *1*, otherwise it is *0*.

```
error_code = IIF(is_error, 1, 0)
```

## INKEY( )

The INKEY( ) function returns an integer representing the most recent key pressed by the operator. It does not halt program execution.

### Syntax

INKEY( )

### Notes

The INKEY( ) function returns an integer between 0 (zero) and 255, corresponding to a value in the IBM Extended Character Set entered from the keyboard.

If there are several characters in the type-ahead buffer, INKEY( ) returns an integer representing the first of the characters entered. This clears the character from the type-ahead buffer.

INKEY( ) returns a zero if no key is pressed.

INKEY( ) returns the control-key equivalent of special keys, such as arrow keys and **Ctrl-Home**.

This function is often used in programs as a condition for branching.

It lets you use arrow and other nonprintable keys within a dBASE III PLUS program.

The keyboard and corresponding INKEY() values are listed in the chart:

| Special Keys | Alternative Keys | INKEY() Returns |
|---|---|---|
| → | Ctrl-D | 4 |
| ← | Ctrl-S | 19 |
| ↑ | Ctrl-E | 5 |
| ↓ | Ctrl-X | 24 |
| Ctrl-→ | Ctrl-B | 2 |
| Ctrl-← | Ctrl-Z | 26 |
| Ins | Ctrl-V | 22 |
| Del | Ctrl-G | 7 |
| Home | Ctrl-A | 1 |
| End | Ctrl-F | 6 |
| PgUp | Ctrl-R | 18 |
| PgDn | Ctrl-C | 3 |
| Ctrl-Home | Ctrl-] | 29 |
| Ctrl-End | Ctrl-W | 23 |
| Esc | Ctrl-Q | 27 |
| Ctrl-PgUp | Ctrl-_ | 31 |
| Ctrl-PgDn | Ctrl-^ | 30 |

Table 6-1 INKEY() values

## Examples

To display the time continuously while waiting for a menu selection, include the following routine in a program:

```
DO WHILE .T.
     *
     <display menu options>
     *
     i = 0
     DO WHILE i = 0
          @ 1,72 SAY TIME()
          i = INKEY()
     ENDDO
     *
     DO CASE
          CASE CHR(i) $ "Qq0"
               RETURN
          CASE CHR(i) $ "Aa1"
               DO <procedure 1>
          CASE CHR(i) $ "Bb2"
               DO <procedure 2>
          CASE CHR(i) $ "Cc3"
               DO <procedure 3>
     ENDCASE
ENDDO
```

## See Also

CHR(), ON KEY, READ, READKEY(), SET TYPEAHEAD,

## INT()

The INT() function converts any numeric expression to an integer. It discards all digits to the right of the decimal point in converting the numeric expression.

## Syntax

INT( < expN > )

## Examples

To convert the number 10.23 to an integer:

```
. ? INT(10.23)
     10
. STORE 10.23 TO x
10.23
. ? INT(x)
          10
. STORE INT(10.23) TO x
   10
```

## See Also

ROUND()

## ISALPHA( )

The ISALPHA( ) function returns a logical True (.T.) if the specified character string begins with an *alpha* character.

### Syntax

ISALPHA( < expC > )

### Usage

ISALPHA( ) determines that a character is an alpha character if it is an upper or lower case letter between a and z.

The ISALPHA( ) function is used for evaluating and manipulating character strings.

### Examples

To test whether a string begins with an alpha character:

```
. ? ISALPHA('abc123')
.T.
. ? ISALPHA('ABC123')
.T.
. ? ISALPHA('123abc')
.F.
```

### See Also

ISLOWER( ), ISUPPER( ), LOWER( ), UPPER( )

## ISCOLOR( )

The ISCOLOR( ) function returns a logical True value (.T.) if a color graphics card is installed in the computer.

### Syntax

ISCOLOR( )

### Usage

The ISCOLOR( ) function enables developers to design applications for both color or monochrome environments.

The response of ISCOLOR( ) depends on whether the system is operating with a color graphics board. Whether you are operating in color actually depends on whether you have a color monitor connected, and whether SET COLOR is ON or OFF.

### Examples

To set the color environment from within a program, depending on whether it is running in color or monochrome mode, you could execute the following commands in a dBASE program:

```
IF ISCOLOR()
   SET COLOR TO GR/B,W/R,GR
ELSE
    SET COLOR TO W+
ENDIF
```

If ISCOLOR( ) returns a True (.T.) value, the program executes a SET COLOR TO command for a color display. Otherwise, it sets the display for monochrome.

### See Also

SET COLOR

## ISLOWER( )

The ISLOWER( ) function returns a logical True value (.T.) if the specified character string begins with a lower case alpha character.

### Syntax

ISLOWER( < expC > )

### Usage

A lower case alpha character is any lower case letter between a and z.

The ISLOWER( ) function is used for evaluating and manipulating character strings. ISLOWER( ) returns a False value (.F.) when either an upper case letter or a non-alpha character is in the first position of the string.

### Examples

To test whether a string begins with a lower case alpha character:

```
. ? ISLOWER('abc123')
.T.
. ? ISLOWER('ABC123')
.F.
. ? ISLOWER('12abc')
.F.
```

### See Also

ISALPHA( ), ISUPPER( ), LOWER( ), UPPER( )

## ISUPPER( )

The ISUPPER( ) function returns a logical True value (.T.) if the specified character string begins with an upper case alpha character.

### Syntax

ISUPPER( < expC > )

### Usage

An upper case alpha character is any upper case letter between A and Z.

The ISUPPER( ) function provides greater flexibility for manipulating character strings.

ISUPPER( ) returns a logical False value (.F.) when either a lower case letter or a non-alpha character is in the first position of the string.

### Examples

To test whether a string begins with an upper case alpha character:

```
. ? ISUPPER('ABC123')
.T.
. ? ISUPPER('abc123')
.F.
. ? ISUPPER('12ABC')
.F.
```

### See Also

ISALPHA( ), ISLOWER( ), LOWER( ), UPPER( )

## LEFT( )

The LEFT( ) function returns a specified number of characters from the specified character string, starting with the first character on the left.

## Syntax

LEFT( < expC > , < expN > )

## Usage

The LEFT( ) function lets you retrieve the first part of a character string. This is the same as defining the SUBSTR function with a starting position of one, and the number of characters to extract with < expN > .

The numeric expression defines the number of characters to extract from the character string. If the numeric expression is zero or negative, dBASE III PLUS returns a null string.

If the numeric expression is greater than the length of the character string, dBASE III PLUS returns the entire string.

## Examples

To extract the first three characters from a character string:

```
. ? LEFT('abcdef',3)
abc
```

## See Also

AT( ), LTRIM( ), RIGHT( ), RTRIM( ), STUFF( ), SUBSTR( ), TRIM( )

## LEN()

The LEN() function returns a numeric value indicating the number of characters in a specified character string.

## Syntax

LEN( < expC > )

## Usage

The length of a null string is zero.

## Examples

To determine the number of characters in the character string "This is a cat":

```
. ? LEN('This is a cat')
      13
. STORE 'This is a cat' TO alpha
This is a cat
. ? LEN(alpha)
      13
```

If the Clients database file contains a character field named Address with a field width of 25, to determine the number of characters in the actual address stored in the field (8307 Santa Anita Blvd):

```
. USE Clients
. ? LEN(Address)
          25

. ? LEN(TRIM(Address))
          21
```

## See Also

TRIM()

## LOG( )

The LOG( ) function returns the natural logarithm of a specified number.

## Syntax

LOG( < expN > )

## Usage

The natural logarithm has a base of **e**. The LOG( ) function returns the exponent in the following equation:

$e^x = y$.

Y is the numeric expression used by the LOG( ) function.

## Examples

To determine the natural log of 2.71828:

```
. ? LOG(2.71828)
      1.00000
. STORE 2.71828 TO x
2.71828
. ? LOG(x)
      1.00000
```

To determine the natural log of 4 * 2:

```
. x = 4
4
. y = 2
2
. ? LOG(x * y)
2.08
```

## See Also

EXP( ), SET DECIMALS, SET FIXED

## LOWER( )

The LOWER( ) function converts upper case letters to lower case letters.

## Syntax

LOWER( < expC > )

## Examples

```
. ? LOWER('THIS IS A NICE DAY')
this is a nice day

. STORE 'THIS IS A NICE DAY' TO niceday
. ? niceday
THIS IS A NICE DAY

. ? niceday = 'this is a nice day'
.F.

. ? LOWER(niceday)='this is a nice day'
.T.

. STORE '123 W. 45th Street, New York, NY' TO street
123 W. 45th Street, New York, NY
. ? LOWER(street)
123 w. 45th street, new york, ny
```

## See Also

ISALPHA(), ISLOWER(), ISUPPER(), UPPER()

## LTRIM( )

**LTRIM( )** The LTRIM( ) function removes leading blanks from a character expression.

**Syntax** LTRIM( < expC > )

**Usage** Use the LTRIM( ) function to remove the leading blanks that result from the STR( ) function.

**Examples** To remove leading blanks from a character string formed from a number:

```
. ? STR(11.95,8,2)
     11.95
. ? LTRIM(STR(11.95,8,2))
11.95
```

**See Also** LEFT( ), RIGHT( ), RTRIM( ), STR( ), SUBSTR( ), TRIM( )

## LUPDATE( )

The LUPDATE( ) function returns the date of the last update of the currently selected database file in a date format.

## Syntax

LUPDATE( )

## Usage

The LUPDATE( ) function specifies when a database file was last updated. The date is displayed in the date format mm/dd/yy. If no database file is in USE, LUPDATE( ) returns a blank date.

## Examples

To make sure that receipts are not entered more than once a day, include the following lines in a data entry program:

```
IF LUPDATE() < DATE()
   Do_entry = '?'
   @ 5,1 SAY 'Receipts were last entered on ' +;
         DTOC(LUPDATE()) + ' Enter now? (Y/N)';
         GET Do_entry PICTURE 'Y'
   READ
   IF Do_entry = 'Y'
      DO <entry procedure>
   ENDIF
ENDIF
```

## See Also

DBF( ), FIELD( ), NDX( ), RECCOUNT( ), RECSIZE( )

## MAX( )

MAX( ) returns the largest value of two numeric expressions.

## Syntax

MAX( < expN1 > , < expN2 > )

## Usage

The MAX( ) function compares two numeric expressions, returning the largest of the two.

## Examples

Using the Travel database file, you can determine whether the values contained in the Cost field are greater than 1500:

```
. USE Travel
. Low_limit = 1500
1500
. LIST NEXT 6 Cost, MAX(Low_limit, Cost)
Record#      Cost  MAX(Low_limit, Cost)
     1   1199.00                1500.00
     2   1378.00                1500.00
     3   3195.00                3195.00
     4   1969.00                1969.00
     5    988.00                1500.00
     6   3195.00                3195.00
. USE
```

To determine the highest Cost in the Travel file, create and run the following program:

```
SET TALK OFF
USE Travel
PUBLIC Last_max
DO WHILE .NOT. EOF()
   Last_max = MAX(Last_max, Cost)
   SKIP
ENDDO
? Last_max
USE
SET TALK ON
```

## See Also

MIN( )

## MESSAGE()

The MESSAGE() function returns the message corresponding to an error detected by dBASE III PLUS.

## Syntax

MESSAGE()

## Usage

The error message character string returned by MESSAGE() can be displayed on the screen, stored to a variable, or otherwise manipulated. See Chapter 7 for a list of all possible error messages in single-user dBASE III PLUS. (For error messages relevant to dBASE III PLUS on a local area network, see Appendix A in *Networking with dBASE III PLUS*.)

## See Also

ERROR()

## MIN( )

MIN( ) returns the smaller value of two numeric expressions.

## Syntax

MIN( < expN1 > , < expN2 > )

## Usage

The MIN( ) function compares two numeric expressions and returns the smaller number.

## Examples

Using the Travel database file, you can determine whether values in the Cost field are lower than $1500.

```
. USE Travel
. High_limit = 1500
1500
. LIST Next 6 Cost, MIN(High_Limit, Cost)
Record#       Cost MIN(High_Limit, Cost)
      1    1199.00                1199.00
      2    1378.00                1378.00
      3    3195.00                1500.00
      4    1969.00                1500.00
      5     988.00                 988.00
      6    3195.00                1500.00
. USE
```

To determine the lowest rental in the Travel database file, you could type in the following program:

```
SET TALK OFF
USE Travel
PUBLIC Last_min
DO WHILE .NOT. EOF()
   Last_min = MIN(Last_min, Cost)
   SKIP
ENDDO
? Last_min
USE
SET TALK ON
```

## See Also

MAX( )

## MOD( )

The MOD( ) function returns the remainder from a division of two numbers. MOD( ) is particularly useful for base number conversions, such as inches to yards.

## Syntax

MOD( < expN1 > , < expN2 > )

## Usage

The MOD( ) function returns a whole number, the *modulus*, which is the remainder of < expN1 > divided by < expN2 > .

< expN1 > is the numeric expression which is being limited
< expN2 > -1 is the upper limit of MOD( ) if < expN2 > is a positive integer.

MOD( ) returns a positive number if < expN2 > is positive and a negative number if < expN2 > is negative.

The modulus formula is:

< expN1 > - floor( < expN1 > / < expN2 > ) * < expN2 >

where floor is a mathematical function that returns the greatest integer less than or equal to its argument. The INT( ) function differs from floor in that INT( ) simply truncates any decimals. Floor is not a dBASE III PLUS function.

## Examples

To determine the MOD( ) of two numbers:

```
. ? MOD(14,12)
 2
. ? MOD(0,32)
 0
. ? MOD(0,-432)
 0
. ? MOD(1,-3)
-2
. ? MOD(-2,3)
 1
. ? MOD(-4,-3)
-1
```

To convert inches to yards:

```
. ? INT(112/36), 'yards', MOD(112,36), 'inches'
        3 yards       4 inches
```

To convert minutes to days, hours, and minutes:

```
. t=36500
36500
. minutes=MOD(t,60)
          20
. h=INT(t/60)
             608
. hours=MOD(h,24)
            8
. days=INT(h/24)
              25
. ? t,"minutes is:",days, "days",hours,"hours",minutes,"minutes"
    36500 minutes is:          25 days          8 hours          20 minutes
```

## MONTH( )

The MONTH( ) function returns a number representing the month from a date expression.

### Syntax

MONTH( < expD > )

### Usage

The date expression is a memory variable, a field, or any function that returns date type data.

### Examples

If the system date is 05/15/84:

```
. ? MONTH(DATE())
5
```

To store the month from the system date to the memory variable Mmonth:

```
. STORE MONTH(DATE()) TO mmonth
5

. ? mmonth
          5

. ? TYPE('mmonth')
N
```

### See Also

CMONTH( ), DAY( ), YEAR( )

## NDX( )

The NDX( ) function returns the names of active index files in the currently selected work area.

## Syntax

NDX( < expN > )

## Usage

The NDX( ) function allows programs to manipulate open index files without knowing the names of the files.

The numeric expression is the position of the open index files in the index file list of the currently selected work area. It *must* be between one and seven.

The order in which the index files were opened determines which index filename is returned for a particular numeric expression. SET ORDER TO has no effect on the results of NDX( ).

If no database file is open or no index file is open in the specified position, dBASE III PLUS returns a null string.

## Examples

To display a list of all open index files, include the following lines in a program:

```
i = 1
Null = ""
DO WHILE Null < NDX(i) .AND. i <=7
   ? NDX(i)
   i = i+1
ENDDO
```

## See Also

DBF( ), FIELD( ), LUPDATE( ), RECCOUNT( ), RECSIZE( ), SET INDEX, SET ORDER TO

## OS()

The OS() function returns the name of the operating system under which dBASE III PLUS is running.

### Syntax

OS()

### Usage

The OS() function lets you make a program portable between DOS and UNIX by determining the operating system under which dBASE III PLUS is running and providing this information to the program.

### See Also

DISKSPACE(), GETENV(), VERSION()

## PCOL()

The PCOL() function returns the current column position on the printer, and helps keep track of printer column positions within programs.

## Syntax

PCOL()

## Usage

You may store the value of PCOL() to a memory variable.

PCOL() may be used for relative addressing. The symbol **$** is used to represent the current row or column position. For example, @ 1,PCOL() + 5 is the same as 1, $ + 5. Both position the printer five columns beyond the current column. Use PCOL() when you want to use a value stored in a memory variable.

When used interactively with SET PRINT ON, the result is always 0.

## Examples

To start printing a specified number of columns from the last printer column position:

```
. SET DEVICE TO PRINT
. @ 1,PCOL() + 5 SAY 'An example of relative addressing'
. SET DEVICE TO SCREEN
```

To determine the current printer position:

```
. ? PCOL()
10
. X = PCOL()
14
```

## See Also

@, COL(), PROW(), ROW()

## PROW()

PROW() is the printer row function. It determines the current row on the printer.

## Syntax

PROW()

## Examples

To determine the current printer position:

```
. ? PROW()
10
. X = PROW()
14
```

To print **An example of relative addressing** five lines below the current printer line:

```
. SET DEVICE TO PRINT
. @ PROW()+5,1 SAY 'An example of relative addressing'
. SET DEVICE TO SCREEN
```

Negative relative addressing is not possible with the PROW() function.

To cause the printer to do a form feed at the end of a page:

```
IF PROW() > 60
   EJECT
ENDIF
```

## See Also

@, COL(), PCOL(), ROW()

## READKEY()

The READKEY() function returns an integer representing the key pressed in order to exit from a full-screen command. READKEY() also indicates whether any data was changed.

### Syntax

READKEY()

### Usage

The READKEY() function helps a program determine what action to take after you exit from one of the full-screen commands: APPEND, BROWSE, CHANGE, CREATE, EDIT, INSERT, MODIFY, or READ.

The following chart shows the integer value of READKEY() for each method of exiting from a full-screen command. Notice that each keypress has two possible codes. If no data is changed, the code is between decimal 0 and decimal 36. If data is updated, the code number increases by 256.

| Non-Updated Code Number | Updated Code Number | Key Pressed | Keypress Meaning |
|---|---|---|---|
| 0 | 256 | **Ctrl**-S, ←, **Ctrl**-H **Backspace** | backward one character |
| 1 | 257 | **Ctrl**-D, →, **Ctrl**-L | forward one character |
| 2 | 258 | **Ctrl**-A, **Home** | backward one word |
| 3 | 259 | **Ctrl**-F, **End** | forward one word |
| 4 | 260 | **Ctrl**-E, **Ctrl**-K, ↑ | backward one field |
| 5 | 261 | **Ctrl**-J, **Ctrl**-X, ↓ | forward one field |
| 6 | 262 | **Ctrl**-R, **PgUp** | backward one screen |
| 7 | 263 | **Ctrl**-C, **PgDn** | forward one screen |
| 12 | -- | **Ctrl**-Q, **Esc** | terminate without save |
| -- | 270 | **Ctrl**-W, **Ctrl-End** | terminate with save |
| 15 | 271 | ↵, **Ctrl**-M | RETURN or fill last record |
| 16 | -- | ↵, **Ctrl**-M | At the beginning of a record in APPEND |
| 33 | 289 | **Ctrl-Home** | menu display toggle |
| 34 | 290 | **Ctrl-PgUp** | zoom out |
| 35 | 291 | **Ctrl-PgDn** | zoom in |
| 36 | 292 | **F1** | HELP function key |

Table 6-2 READKEY() Coding

## Examples

To determine if the contents of a memory variable were altered and to REPLACE a field if changes were made, include the following in a program:

```
IF READKEY() = 270                    && Ctrl-END or Ctrl-W was pressed
   REPLACE Name WITH Mname
ENDIF
```

## See Also

INKEY(), ON KEY, READ

## RECCOUNT()

RECCOUNT() returns the number of records in the currently selected database file.

### Syntax

RECCOUNT()

### Usage

Using the RECCOUNT() function is a faster equivalent to issuing a GO BOTTOM followed by RECNO() on an unindexed file.

If no file is in use, RECCOUNT() returns a zero.

RECCOUNT() includes *all* records, even if SET DELETED is ON or SET FILTER is in effect.

### Tips

Use RECCOUNT() with RECSIZE() and DISKSPACE() in application programs that automatically back up database files. This enables you to check if there is sufficient space on the disk for the backup file. See the example in the RECSIZE() entry.

### Examples

To find the number of records in the Travel file:

```
. USE Travel
. ? RECCOUNT()
      49
. USE
```

### See Also

DBF(), DISKSPACE(), FIELD(), NDX(), RECSIZE()

## RECNO()

The RECNO() function returns the current record number of the selected file.

## Syntax

RECNO()

## Defaults

If there are no records in a database file, RECNO() = 1 and EOF() is True (.T.). If no database file is in use, RECNO() = 0.

## Record Pointer

If the record pointer moves past the last record in the file (and EOF() is True), RECNO() returns a value which is one more than the number of records in the file.

If the record pointer moves backward to before the first record in the file (and BOF() is True), RECNO() is 1.

## Examples

To display the current record number:

```
. USE Clients
. ? RECNO()
1
. SKIP -1
Record no.      1
. ? BOF()
.T.
. GO BOTTOM
. ? RECNO()
          49
. SKIP
Record No.      50
. ? EOF()
.T.
```

You may store RECNO() to a memory variable:

```
. STORE RECNO() TO recnbr
         50
```

## RECSIZE( )

The RECSIZE( ) function returns the size of a record in the currently selected database file.

## Syntax

RECSIZE( )

## Usage

If no file is in use, RECSIZE( ) returns a zero.

The RECSIZE( ) function enables a program to manipulate files which are larger than available disk space.

## Programming Notes

Use RECSIZE( ) with RECCOUNT( ) and DISKSPACE( ) in application programs that automatically back up database files. This enables you to check if there is sufficient space on the disk for the backup file. You also must know the header size, which is calculated as follows:

32 * <number of fields> + 35.

## Examples

To copy a large file named File from a hard disk to several floppy disks on drive B, you can use the following routine in a program file:

```
USE File
SET DEFAULT TO B
DO WHILE .NOT. EOF()
    WAIT 'Insert new disk in drive B, and press a key...'
    COPY NEXT (DISKSPACE() - <header size>)/RECSIZE() ;
      TO Backup
    SKIP
ENDDO
USE
```

## See Also

DBF( ), DISKSPACE( ), FIELD( ), NDX( ), RECCOUNT( )

## REPLICATE( )

REPLICATE( ) repeats a character expression a specified number of times.

### Syntax

REPLICATE( < expC > , < expN > )

### Usage

The character expression defines what character string to repeat. The numeric expression tells REPLICATE( ) how many times to repeat the character. REPLICATE( ) will only use the integer portion of the evaluated numeric expression.

The resulting character expression must not exceed 254 characters. Therefore, the numeric expression must be a number less than 254 divided by the number of characters in < expC > .

The REPLICATE( ) function lets you produce a horizontal bar graph and is useful whenever you want to repeat a character.

### Examples

To create a bar graph that enables you to see the commission rate in the Percent numeric field:

```
. USE Tours
. LIST NEXT 5 Percent, REPLICATE('*',Percent)
Record#  Percent REPLICATE('*',Percent)
      1   10.000  **********
      2    7.500  *******
      3    5.000  *****
      4   10.000  **********
      5    7.500  *******
```

## RIGHT()

The RIGHT() function returns a specified number of characters from a character string, counting from the last character in the string.

## Syntax

RIGHT( < expC > , < expN > )

## Usage

The RIGHT() function allows you to extract a substring of characters from a character string. The numeric expression defines the number of characters to extract from the character string, starting from the last character on the right in the character string. If the numeric expression is zero or negative, RIGHT() returns a null string.

If the numeric expression is greater than the length of the character string, RIGHT() returns the entire string.

## Examples

To extract the last three characters from a character expression:

```
. ? RIGHT('abcdef',3)
def
```

## See Also

LEFT(), LTRIM(), RTRIM(), STUFF(), SUBSTR(), TRIM()

## ROUND()

The ROUND() function rounds off numbers to a specified number of decimal places.

## Syntax

ROUND( < expN1 > , < expN2 > )

## Usage

< expN1 > is the number or numeric expression you want to round off. < expN2 > is the number of decimal places you want to retain. If < expN2 > is negative, ROUND() returns a rounded whole number.

## Examples

```
. ? ROUND(14.746321,2)
14.750000

. ? ROUND(17.321111,3)
17.321000

. STORE 10.7654321 TO x
10.7654321

. ? ROUND(x,0)
11.0000000

. ? ROUND(x,1)
10.8000000
. ? ROUND (14911,-3)
15000
```

## See Also

INT()

## ROW()

The ROW() function returns the row number of the current cursor position.

## Syntax

ROW()

## Usage

ROW() is most useful in programs to keep track of rows used for relative addressing of screen locations.

## Examples

You can determine the current cursor position in the following manner, although the actual response depends on where you are on the screen:

```
. ? ROW()
10

. x = ROW()
. ? x
14
```

In a command file, to display "This is an example of relative addressing" five lines below the current screen cursor location (the cursor must be above row 19):

```
@ ROW()+5,1 SAY 'This is an example of relative addressing'
```

## See Also

@, COL(), PCOL(), PROW(),

## RTRIM( )

**Syntax** RTRIM( < expC > )

**Usage** The RTRIM( ) function removes trailing blanks from a character string.

**See Also** LEFT( ), LTRIM( ), RIGHT( ), TRIM( )

## SPACE( )

SPACE( ) generates a character string consisting of a specified number of blanks.

## Syntax

SPACE( <expN> )

## Usage

The largest number of spaces that you can specify with the SPACE( ) function is 254.

## Examples

To create a memory variable consisting of 20 blank spaces, you would type:

```
. STORE SPACE(20) TO blanks

. ? '*' + blanks + '*'
*                    *
```

## SQRT( )

The SQRT( ) function returns the square root of a positive number.

## Syntax

SQRT( < expN > )

## Usage

SQRT( ) returns a square root value with the same number of decimal places as the number, or the specified number of decimal places specified by SET DECIMALS TO, whichever is larger.

## Examples

To determine the square root of 4:

```
. ? SQRT(4)
2.00

. ? SQRT(2*2)
  2.00

. STORE 4.000 TO number
4.000
. ? SQRT(number)
        2.000
```

## See Also

SET DECIMAL, SET FIXED

## STR()

The STR() function converts a number to a character string.

### Syntax

STR( < expN > [, < length > ] [, < decimal > ])

### Defaults

If the length is not specified, the string length is 10 characters. If no decimal places are specified, the number is rounded to an integer.

### Usage

The length you specify is the total number of characters in the string created by STR(), including, if applicable, the decimal point, minus sign, and the number of decimal places.

The number you specify for Decimal is the total number of decimal places to output.

If you specify a smaller length than there are digits to the left of the decimal in the numeric expression, dBASE III PLUS returns asterisks in place of the number.

If you specify fewer decimals than are in the numeric expression, dBASE III PLUS truncates the result to the specified number of decimal places.

### Examples

To display the number 11.14 as a character string:

```
. ? STR(11.14,5,2)
11.14
```

To display the number 11.14 * 10 as a character string:

```
. STORE 11.14 TO x
11.14
. STORE STR(x*10,5) TO y
  111
. ? y
  111
. STORE STR(x*10,5,1) TO y
111.4
. ? y
111.4
. STORE STR(x*10,5,2) TO y
111.4
. ? y
111.4
. STORE STR(x*10,6,2) TO y
111.40
. ? y
111.40
```

## See Also

VAL()

## STUFF( )

The STUFF( ) function replaces a portion of a character string with another specified character string.

### Syntax

STUFF( < expC1 > , < start position > , < number of characters > , < expC2 > )

### Usage

Character expression < expC1 > is the character string you want to alter.

Character expression < expC2 > is the character string you want to substitute for a specified number of characters in < expC1 > .

The start position is a numeric expression that indicates where you want to begin the replacement in < expC1 > .

The number of characters is a numeric expression that indicates how many characters you want to remove from the original string. If the number of characters is zero, the second character expression is inserted, and no characters are removed from < expC1 > .

If character expression < expC2 > is a null string, dBASE III PLUS removes the specified number of characters from character expression < expC1 > without adding characters.

The new string will not be the same size as the original string if the specified number of characters is different than the actual number of characters in < expC2 > .

### Examples

To replace a string:

```
. ? STUFF('abc',2,1,'xyz')
axyzc
. ? STUFF('abc',2,1,"")
ac
. ? STUFF('abc',2,0,'xyz')
axyzbc
```

To make a replacement in the Address field of the Clients database file:

```
. new_street = 'Lithe'
Lithe
. USE Clients
. GO 5
. ? Address
1960 Lindley Ave.
. REPLACE Address WITH STUFF(Address,6,7, new_street)
      1 record replaced
. ? Address
1960 Lithe Ave.
. CLOSE DATABASES
```

## See Also

LEFT(), RIGHT(), SUBSTR()

## SUBSTR( )

The SUBSTR( ) function extracts a specified number of characters from a character string.

### Syntax

SUBSTR( < expC > , < starting position > [, < number of characters > ])

### Defaults

If the number of characters is omitted, the function returns a substring that begins with the starting position and ends with the last character of the original character string.

### Usage

If the number of characters entered is greater than the number of characters between the starting position and the end of the original character string, the function returns a substring that begins at the specified starting position and ends with the last character of the original character string. The starting position must be positive.

### Examples

To extract the substring "59" from the string "1958 1959 1960":

```
. ? SUBSTR('1958 1959 1960',8,2)
59

. STORE 'English Spanish Italian German French' TO language
English Spanish Italian German French
. ? SUBSTR(Language,9,7)
Spanish

. STORE SUBSTR(Language,17,7) TO l1
Italian
. ? l1
Italian
```

### See Also

AT( ), LEFT( ), RIGHT( ), STR( ), STUFF( )

## TIME( )

The TIME( ) function returns the system time as a character string in the format hh:mm:ss.

## Syntax

TIME( )

## Usage

To use TIME( ) in calculations, convert the value returned to a numeric value using SUBSTR( ) and VAL( ).

## Examples

```
. ? TIME()
21:22:23
. STORE TIME() TO mtime
21:22:23
. ? mtime
21:22:23

* Time calculation program
Start_time = TIME()
Tally = 0
DO WHILE Tally < 100
    Tally = Tally + 1
ENDDO
End_time = TIME ()
Old_time = VAL(SUBSTR(Start_time, 1, 2)) * 3600 +;
           VAL(SUBSTR(Start_time, 4, 2)) * 60 +;
           VAL(SUBSTR(Start_time, 7, 2))
New_time = VAL(SUBSTR(End_time, 1, 2)) * 3600 +;
           VAL(SUBSTR(End_time, 4, 2)) * 60 +;
           VAL(SUBHSTR(End_time, 7, 2))
Difference = New_time  - Old_time
? "Elapsed time was",LTRIM(STR(Difference)), "seconds."
```

## TRANSFORM( )

The TRANSFORM( ) function allows PICTURE formatting of character and numeric data without using the @...SAY command.

## Syntax

TRANSFORM( < exp1 > , < expC > )

## Defaults

The result of the TRANSFORM( ) function is always a character, even if the first expression specifies a number.

## Usage

The character expression < expC > defines the PICTURE format of < exp1 > , which may specify a character string or a number.

The TRANSFORM( ) function lets you use a PICTURE format with characters and numbers by use of ?, ??, DISPLAY, LABEL, LIST, and REPORT.

For information about PICTURE options, refer to the @ command.

## Examples

To display the twenty-character wide character field, Lastname, from the Travel database file, providing a space between each character for emphasis, you could type the following:

```
. USE Travel
. LIST NEXT 4 TRANSFORM(Lastname,'@R X X X X X X X X X X X X X X X X X X X X')
Record#  TRANSFORM(Lastname, '@R X X X X X X X X X X X X X X X X X X X X')
       1 B u c k m a n
       2 L i s b o n n
       3 B i c k s b y
       4 G a r n e t t
```

To display the ten-character wide numeric field, Cost, from the Travel file, inserting commas, you could type the following:

```
. LIST NEXT 4 TRANSFORM(Cost,'#,###,###.##')
Record#  TRANSFORM(Cost,'#,###,###.##')
       4  1,969.00
       5    988.00
       6  3,195.00
       7    988.00
```

## See Also

@...PICTURE

## TRIM( )

The TRIM( ) function removes all trailing blanks from a character string.

## Syntax

TRIM( < expC > )

## Usage

TRIM( ) is followed either by a comma (,) or a plus sign ( + ) when used in a list of more than one field. A comma following TRIM( ) inserts one blank space before the listing of the next field in the list. A plus sign following TRIM( ) does not add any spaces before the next field in the list is displayed.

If you use TRIM( < field name > ) with either LIST or DISPLAY, the trailing blanks are not removed. Instead, the field contents are trimmed, but the field widths remain the same.

## Examples

The Clients database file contains the fields Lastname and Address used in this example.

```
. USE Clients
. ? Lastname,  Address
Buckman              8307 Santa Anita Blvd
```

The address is separated from the name by the number of trailing blanks in the Lastname field. To eliminate the blank spaces between the name and the address:

```
. ? TRIM(Lastname),Address
Buckman 8307 Santa Anita Blvd
```

The single space now separating the name from the address is caused by the comma in the command. If you used a plus ( + ) sign, there would be no spaces between the name and address.

```
. ? TRIM(Lastname) + Address
Buckman8307 Santa Anita Blvd
```

In the following examples, the first and last names are trimmed, but the trailing blanks follow. The field widths remain the same.

```
. DISPLAY TRIM(Lastname),Address
Record#    TRIM(Lastname)      Address
      1   Buckman             8307 Santa Anita Blvd.

. DISPLAY TRIM(Lastname) + ',' +TRIM(Firstname),Address
Record#   TRIM(Lastname) + ',' +TRIM(Firstname)       Address
      1   Buckman,Claire                            8307 Santa Anita Blvd
```

The number of spaces after 'Claire' equals the number of trailing blanks in the Lastname and Firstname fields, plus one blank space for the commas. The names plus the spaces that follow equal the combined width of the Lastname and Firstname fields.

## See Also

LTRIM(), RTRIM()

## TYPE( )

The TYPE( ) function evaluates an expression and returns a single upper case character that indicates whether the expression is character, numeric, logical, memo, or undefined. It is a test for the existence of a variable and/or the validity of an expression.

### Syntax

TYPE( < expC > )

### Usage

The TYPE( ) function always returns an upper case letter.

The possible TYPE( ) values are:

Character = C
Numeric = N
Logical = L
Memo = M
Undefined = U

### Examples

To determine if the memory variable Test exists (note that the variable name is delimited by quotes unless it refers to another variable):

```
. ? TYPE('test')
U
```

U indicates that the memory variable does not exist. To test for a memory variable which does exist:

```
. STORE 'testing 1 2 3' TO test
testing 1 2 3
. ? TYPE('test')
C
```

C indicates that the memory variable does exist and that it is a character type memory variable.

To evaluate the expression "test/100" and determine if the expression is valid (this test can be used to avoid syntax errors):

```
. STORE 'hello' TO test
hello
. ? TYPE('test/100')
U
```

TYPE( ) returns a U since a character string cannot be divided by a number.

```
. STORE 5 TO x
5
. STORE 7 TO y
7
. ? TYPE('x*y/4')
N
```

TYPE( ) returns N since the operation evaluated creates a valid expression.

To test an expression for validity, compare the result of the TYPE( ) function to the upper case letter representing the expression type:

```
. ? TYPE('(1 + 2) = 5') = 'L'
.T.
```

When the delimiters are omitted, TYPE( ) evaluates a variable that is the content of the named variable. In the following instance, the expression 'Sample' does not need to be delimited:

```
. STORE 5 TO cat
5
. STORE 'cat' TO sample
cat
. ? TYPE(sample)
N
```

TYPE( ) returns an N since 'cat', a numeric variable, is referred to in Sample.

## UPPER( )

The UPPER( ) function converts lower-case letters to upper-case letters.

## Syntax

UPPER( < expC > )

## Examples

```
. ? UPPER('This is a nice day')
THIS IS A NICE DAY
. STORE 'This is a nice day' TO niceday
This is a nice day
. ? niceday
This is a nice day
```

To display this variable in upper case:

```
. ? UPPER(niceday)
THIS IS A NICE DAY

. ? niceday='THIS IS A NICE DAY'
.F.

. ? UPPER(niceday)='THIS IS A NICE DAY'
.T.
```

To change a character in the account number (Ac) field of a database file to upper case:

```
. GO TOP
. LIST AC
Record  #    AC
        1    a03
        2    a39
        3    a47
        4    a21
. REPLACE ALL Ac WITH STUFF(Ac, 1, 1, UPPER(SUBSTR(Ac, 1, 1)))
4 records replaced
. LIST AC
RECORD  #    AC
        1    A03
        2    A39
        3    A47
        4    A21
```

## See Also

ISLOWER( ), ISUPPER( ), LOWER( )

## VAL()

The VAL( ) function converts numbers that are defined as characters into a numeric expression.

## Syntax

VAL( < expC > )

## Usage

If the expression consists of leading non-numeric characters other than blanks, dBASE III PLUS returns a value of zero.

VAL( ) displays the decimal portion of a number as designated by SET DECIMALS.

The VAL( ) function operates from left to right, converting characters to numeric values until a text character is encountered. Leading blanks are ignored if the argument contains both numeric and non-numeric characters. The leading numeric characters will be converted to a numeric value. Trailing blanks are treated as text characters and, when encountered, terminate the conversion process.

## Examples

The value of a non-numeric character sequence in VAL( ) is zero:

```
. ? VAL('ABC')
  0.00

. SET DECIMALS TO 1
. ? VAL('123.45')
    123.5
. STORE '123.45' TO x
123.45
. STORE VAL(x) TO y
   123.5
. ? y
       123.5

. DISPLAY MEMORY
x           pub C '123.45'
y           pub N        123.5 (        123.45000000)
   2 variables defined,     17 bytes used
 254 variables available, 5983 bytes available
```

VAL() displays 123.5, but STOREs 123.45 to memory.

```
. SET DECIMALS TO 0
. ? STR(Y,6,2)
123.45
. ? 0.00 + Y
      123.45
. ? VAL('A=1231)
        0
. ? VAL('123=A1)
      123
```

## See Also

SET DECIMALS, SET FIXED, STR()

## VERSION()

The VERSION() function returns a character string containing the number of the dBASE III PLUS version in use.

### Syntax

VERSION()

### Usage

The VERSION() function returns the version number that appears in the dBASE III PLUS sign-on message.

This function is useful in applications that utilize version-specific features.

### Examples

To return the version:

```
. ? VERSION()
dBASE III PLUS VERSION 1.0
```

### See Also

DISKSPACE(), GETENV(), OS()

## YEAR( )

The YEAR( ) function returns the numeric value of the year from a date expression. The result is always a four-digit number.

### Syntax

YEAR( < expD > )

### Usage

You can INDEX a date and character together. For example, if you want to INDEX a date field chronologically and have the last names in alphabetical order for each date:

INDEX ON STR(YEAR( < date > ),4) + STR(MONTH( < date > ),2) + STR(DAY( < date > ),2) + Last_name

### Examples

If the system date is 05/15/84:

```
. ? YEAR(DATE())
1984
```

To store the year from the system date to the memory variable Myear:

```
. STORE YEAR(DATE()) TO myear
1984
. ? myear
   1984

. ? TYPE('myear')
N
```

To increase the current system year by 56 and store the result year to the numeric memory variable Newyear:

```
. STORE YEAR(DATE()) + 56 TO newyear
2040
```

# Chapter 7
## ERROR MESSAGES AND THEIR EXPLANATIONS

Below is a list of error messages for dBASE III PLUS.

The messages are arranged alphabetically. To the right of the error messages are numbers in square brackets. These are the numbers returned by the ERROR( ) function for the corresponding error. If no error message number is listed, the ERROR( ) function does not return a number.

Error messages that are specific to dBASE III PLUS on a local area network can be found in *Networking with dBASE III PLUS*.

**A DBF file in view is not in current directory**
MODIFY VIEW cannot modify the stored view because a database file in the view is not located in the current directory.

**ALIAS name already in use.** **[24]**
You attempted to USE a database file that is already open, has the same ALIAS, or has a name or ALIAS within the range A through J. dBASE III PLUS uses A through J for default ALIASes.

**ALIAS not found.** **[13]**
You attempted to either SELECT a database area that is outside the range of A through J, or use an ALIAS that has not been defined.

**ALTERNATE could not be opened**
The ALTERNATE file specified in Config.db could not be opened because of insufficient directory space or an invalid filename.

**At least one file must be in use for this operation**
In CREATE/MODIFY VIEW, you attempted to access the Options Menu before selecting a database file in Set Up.

**Beginning of file encountered.** **[38]**
You positioned the record pointer before the first record in the file.

**Cannot append in column order.** **[147]**
You tried to APPEND FROM a Multiplan spreadsheet (SYLK) with rows not in ascending order. Check that the spreadsheet rows are in the correct order before reissuing the command.

**Cannot display using an invalid filter. Leave Display option.**
In CREATE/MODIFY QUERY, you selected the Display option with an incomplete or invalid query condition. Check that you have opened all the necessary database files and correctly entered all conditions in the filter.

**Cannot erase a file which is open.**
You attempted to erase an open file. Close the file before erasing.

**Cannot JOIN a file with itself.** **[139]**
You named the active database file as the file to JOIN WITH. You need to name another database file in USE.

**Cannot save an invalid filter.**
In CREATE/MODIFY QUERY, you tried to save an incomplete or invalid query condition. Check that you have opened all the necessary database files and correctly entered all conditions in the filter.

**Cannot select requested database.** **[17]**
The work area you SELECTed is outside the range of one through ten.

**Cannot write to read-only file.** **[111]**
You attempted to write to a file that has the read-only attribute set. This attribute is set through the operating system.

**Can't go to a higher column**
In CREATE/MODIFY REPORT, you pressed **PgDn** to add a new column, but you are already on the last column and cannot add another.

**Can't go to a lower column**
In CREATE/MODIFY REPORT, you pressed **PgUp** to add a new column when you are on the first column.

**Can't have subgroups without groups**
In CREATE/MODIFY REPORT, you attempted to define a subgroup when there is no group defined.

**Command not recognized!**
The command you entered while in HELP does not exist.

**CONTINUE without LOCATE.** **[42]**
You attempted to execute a CONTINUE command without first executing a LOCATE command.

**Cyclic relation.** **[44]**
You created an endless cycle by SETting RELATION INTO a database that relates back to the currently selected database. An example of this situation occurs when you have three databases (A, B, and C), and A relates to B, B to C, and C to A. SET RELATION TO without any parameters disconnects the relation in the currently selected work area.

**Data Catalog has not been established** **[122]**

You attempted to use the catalog query (?) clause as part of a command line when there was no catalog engaged. Use SET CATALOG TO to open a catalog before using the catalog query clause in command syntax.

**Data type mismatch.** **[9]**

You attempted an operation with conflicting data types. Some of the conditions that cause this error message to occur are summarized below under the headings Expressions, REPLACE, SEEK, and SORT.

**Expressions**

You attempted to create an expression which contains different data types. You need to correct the expression so it can be evaluated to a result of a single data type.

**REPLACE**

You tried to REPLACE a field with an expression of a different data type.

**SEEK**

You used the SEEK command with an expression that does not evaluate as the same data type as the active index key.

**SORT**

You attempted to SORT on a logical or memo field.

**Database is encrypted.** **[131]**

In single-user dBASE III PLUS, you attempted to USE a database file that was encrypted using dBASE III PLUS on a local area network.

**Database is not indexed.** **[26]**

You attempted to execute a FIND, SEEK, or UPDATE command on a database file without having an index file open, or attempted to SET RELATION to a database file that is not indexed.

**.DBT file cannot be opened.** **[41]**

You attempted to USE a database file containing a memo field when dBASE III PLUS cannot find the .dbt file containing the memo field text.

**Disk full when writing file: <filename>**

You attempted to write to a disk that has no more space available.

**<filename> does not exist. Deleted from catalog.**
You included a database file in a catalog that is not on the disk. SET CATALOG TO checks each file in the specified catalog to make sure it exists. If the file is not in the location specified in the catalog, dBASE III PLUS deletes the filename from the catalog.

**DOs nested too deep.** **[103]**
You can only have a maximum of 20 nested DO commands.

**Empty database file.**
In CREATE/MODIFY QUERY, you tried to Display records from an empty database file.

**Empty structure will not be saved.**
This is a warning message. You are exiting from CREATE or MODIFY STRUCTURE when the database file structure contains no fields. The database file will not be saved.

**End of file encountered.** **[4]**
You attempted to move the record pointer past the end of the database in USE. There are four main conditions that cause this error message to be displayed. These conditions are summarized below under the headings APPEND FROM, UPDATE FROM, USE; FIND, SEEK; MODIFY LABEL/REPORT, LABEL/REPORT FORM; and SKIP.

**APPEND FROM, UPDATE FROM, USE**

You opened a database containing an incomplete structure.

**FIND, SEEK**

You executed a FIND or SEEK command on a corrupted or incomplete index.

**MODIFY LABEL/REPORT, LABEL/REPORT FORM**

You opened a LABEL/REPORT FORM file that has an incomplete structure, or the database file structure is corrupted.

**SKIP**

You SKIPped past the last record in the database file.

**Error in configuration value.**
You specified invalid or incorrect values in the Config.db file. For example, you may have defined a function key name or number that does not exist.

**Error in the HELP system.**
You asked for help while in The Assistant, and the Assist.hlp file is corrupted. You cannot use the help facility in The Assistant until you copy the Assist.hlp file from your master dBASE III PLUS system disk to your Backup working copy.

**Error reading database file.**
In CREATE/MODIFY QUERY, you tried to Display records from a corrupt database file. Repeat the command using a backup file.

**Exceeded report print width.**
The width entered in the Columns pull-down menu causes the report width to exceed the page width specified in the Options Menu in CREATE/MODIFY REPORT.

**Excess header lines lost (press SPACE).**
You attempted to enter a report form heading exceeding the maximum of four lines by using semicolons with the CREATE/MODIFY REPORT command. Characters in the extra lines will not appear in the report.

*****Execution error on + : Concatenated string too large. [77]**
You attempted to concatenate a string with a resulting length greater than the maximum of 254 characters.

*****Execution error on − : Concatenated string too large. [76]**
You attempted to concatenate a string with a resulting length greater than the maximum of 254 characters.

*****Execution error on ^ or ** : Negative base, fractional exponent. [78]**
You attempted to raise a negative number to a power between zero and one.

*****Execution error on CHR( ) : Out of range. [57]**
You used a number less than 0 or greater than 255 as the argument in the CHR( ) function.

*****Execution error on LOG( ) : Zero or negative. [58]**
You used a number less than or equal to zero as the argument in the logarithm function.

*****Execution error on NDX( ): Invalid index number. [87]**
You used a number less than one or greater than seven as the argument in the NDX( ) function.

*****Execution error on REPLICATE( ): String too large. [88]**
You attempted to create a string larger than 254 characters.

*****Execution error on SPACE( ) : Negative.** **[60]**
You used the SPACE( ) function with a negative argument.

*****Execution error on SPACE( ) : Too large.** **[59]**
You used the SPACE( ) function with an argument greater than 254.

*****Execution error on SQRT( ) : Negative.** **[61]**
You used a negative number as the argument in the square root function.

*****Execution error on STORE : String too large.** **[79]**
You attempted to create a memory variable with more than the maximum of 254 characters.

*****Execution error on STR( ) : Out of range.** **[63]**
You specified a STR expression in which the length parameter exceeds 19 digits or the decimal parameter exceeds the length minus two.

*****Execution error on STUFF( ) : String too large.** **[102]**
You specified a STUFF expression which would have resulted in a string that exceeds the allowed maximum of 254 characters.

*****Execution error on SUBSTR( ) : Start point out of range.** **[62]**
You attempted to extract a substring from a character string where the starting position argument of the substring function is greater than the length of the string.

**^^ Expected ON or OFF.**
You set a Config.db ON/OFF parameter to something other than ON or OFF.

**Field name already in use.**
You attempted to use the same field name more than once while CREATing or MODIFYing the STRUCTURE of a database.

**Field not found.**
While in BROWSE, you attempted to FREEZE a field that does not exist.

**Field type must be C, N, D, L or M.**
While in CREATE or MODIFY STRUCTURE, you attempted to declare a field as a type other than character, numeric, date, logical, or memo.

**Fields list too complicated.** **[141]**
You defined a fields list that dBASE III PLUS cannot use. You need to reduce the number of fields specified in the fields list.

**File already exists.** **[7]**
You attempted to RENAME a file TO the name of a file that already exists on the disk.

**File does not exist.** **[1]**
You attempted to open or erase a file that does not exist.

**File is already open.** **[3]**
You attempted to open a file that is already open.

**File is not accessible.** **[29]**
You attempted to create a disk file when the directory was full or when the filename contained one or more of the following invalid characters: + " ? * [ ].
On a network, you attempted to create a file in a read-only directory.

**File not cataloged since SET CATALOG was OFF when the active database was USEd.**
You opened a database file while SET CATALOG was OFF. Always open database files with SET CATALOG ON so the program knows which auxiliary files, such as index, format, or query files, to associate with the database file.

**File too large, some data may be lost**
You attempted to edit a large text file or memo field with the dBASE III PLUS word processor. The maximum text file size that the built-in word processor can handle is 5,000 characters.

**File was not LOADed.** **[91]**
You attempted to LOAD a file that cannot be LOADed (zero length or not available), or you attempted to CALL or RELEASE a MODULE that has not been LOADed.

**Format files must be in row major order to be exported.**
Coordinates used with the @ commands in your format file are not in ascending order. For example, you have an @ command with a row coordinate that is smaller than the row coordinate in the preceding @ command.

**Full screen view allows only one continuous relation chain.**
In CREATE/MODIFY VIEW, the Relate option requires that you select a filename that appears in the relation chain at the bottom of the screen.

**HELP text not found**
You asked for HELP from the dot prompt, and the Help.dbs file is not on the same disk or directory as Dbase.ovl.

**Illegal character data length**
You specified a structure-extended file in CREATE FROM that has a length specification for a field that is not within the range of 1 through 254.

**Illegal data length**

You specified a character-type field length in CREATE or MODIFY STRUCTURE that is not within the range of 1 through 254.

**Illegal decimal length**

You specified a numeric-type field length with the number of decimal places less than 0, greater than 15, or greater than the total numeric field width minus 2.

**Illegal field name**

You specified a structure-extended file in CREATE FROM that contains a specification for a field name with invalid characters.

**Illegal field type**

You specified a structure-extended file in CREATE FROM that contains a specification for data type that is other than C, N, D, L, or M.

**Illegal numeric data length**

You specified a numeric-type field length that is either less than 1 or greater than 19.

**Illegal value.** **[46]**

Using the @ command to draw a box, you specified a TO row or column coordinate that was smaller than the first row or column coordinate. This message can also occur with the full-screen SET command and SET TO commands if you specify an invalid number.

**Improper data type in group expression.**

You ran a REPORT FORM that has a logical or memo data type specified in a group expression.

**Improper data type in subgroup expression.**

You ran a REPORT FORM that has a logical or memo data type specified in a subgroup expression.

**Index expression is too big (220 char maximum).** **[112]**

The key expression you INDEXed ON exceeds the maximum length of 220 characters. This limitation refers to the number of characters in the key expression, rather than the actual length of the resulting key. For example, the following key expression is 22 characters long:

```
INDEX ON First_name + Last_name TO Alpha
```

That is, 10 characters for First_name, 3 characters for the plus sign and spaces on either side, and 9 characters for Last_name. The actual key length may be larger or smaller, depending on the length of the First_name and Last_name fields in the database file structure.

**Index file does not match database.** **[19]**

You attempted to open an index file whose key expression cannot be related to the database in USE.

**Index interrupted. Index will be deleted if not completed.**

You pressed **Esc** during INDEX ON or REINDEX. This error message gives you the opportunity to continue or abandon the indexing process.

**Index is too big (100 char maximum).** **[23]**

The result of the key expression you INDEXed ON exceeds the maximum length of 100 characters. This limitation refers to the actual length of the key when the expression is evaluated. For example, the following key length is 60 characters long if both key fields have a width of 30 in the database file structure.

INDEX ON First-name + Last-name TO Alpha.

**Insert impossible.**

In CREATE/MODIFY REPORT, you attempted to insert a field when the maximum of 24 fields were already present.

**Insufficient dynamic memory for contents window.**

Your computer does not have enough free memory to display the Contents pull-down menu in CREATE/MODIFY LABEL.

**Insufficient memory.** **[43]**

You attempted to load dBASE III PLUS on a machine with less than 256K of available RAM. Some of the conditions that cause this message to be displayed are summarized below:

**@...GET**

You exceeded the maximum number of 128 GETs before issuing a CLEAR GETS, CLEAR or READ command.

**RUN, TEDIT, WP**

You attempted to RUN an external program with insufficient memory. You must have enough extra memory to load Command.com and the other software program.

**LOAD**

You attempted to LOAD a file larger than the amount of available memory.

**Insufficient memory to display fields.**

Your computer does not have enough free memory to display the requested field pull-down menu.

**Internal error: CMDSET( ):**
There is a problem with the dBASE III PLUS system. Make a new working copy of dBASE III PLUS from your master disk. If you still get the error, contact Ashton-Tate Software Support.

**Internal error: EVAL work area overflow**
You attempted to use an expression that cannot be evaluated. Check the expression you entered to make sure the result produces a single data type.

**Internal error: Illegal opcode**
There is a problem with the dBASE III PLUS system. Make a new working copy of dBASE III PLUS from your master disk. If you still get the error, try the following steps:

1. Run the DOS CHKDSK utility and check for corrupted data.
2. Remove RAM resident software.
3. Check that you have the correct Config.sys file.
4. Check database (.dbf) files for damaged data.
5. Erase and re-create index files.

If you still get the error, contact Ashton-Tate Software Support.

This error might also occur during execution of an extremely complex expression.

**Internal error: Unknown command code:**
There is a problem with the dBASE III PLUS system. Make a new working copy of dBASE III PLUS from your master disk. If you still get the error, contact Ashton-Tate Software Support.

***** INTERRUPTED *****
You terminated a command or a command file by pressing the **Esc** key.

**Invalid date (press SPACE)**
You attempted to enter an invalid date into a date-type field or memory variable. Press the **Spacebar** to exit this condition.

**Invalid DIF Character. [118]**
You attempted to APPEND FROM a DIF file that contains an invalid character. This could also be caused by the presence of a control character in the DIF file.

**Invalid DIF File Header [115]**
You attempted to APPEND FROM a DIF file that has an invalid file header.

**Invalid DIF Type Indicator** **[117]**
You attempted to APPEND FROM a DIF file that contains an incorrect data type indicator.

**Invalid DIF vector — DBF field mismatch.** **[116]**
You tried to APPEND FROM a DIF file containing data conflicting with its header record.

**Invalid function argument.** **[11]**
The expression you used as the argument to the function is not the correct input data type for that function.

**Invalid function name.** **[31]**
You specified a function that dBASE III PLUS cannot recognize.

**Invalid index number.** **[106]**
You attempted to SET ORDER TO a position in the index file list that is not occupied or that is out of the range of 0 (zero) to 7.

**Invalid operator.** **[107]**
In attempting to specify a dBASE expression, you used an operator that is invalid for the data types used in the expression (for example, using the "/" operator to connect two character strings).

**Invalid printer port.** **[123]**
You attempted to SET PRINTER TO an operating system device that does not exist. Check your operating system manual for a list of valid printer device names.

**Invalid printer redirection.** **[124]**
You attempted to use the SET PRINTER TO \\ syntax in order to spool output to the network printer, and the printer you specified does not exist.

**Invalid SYLK File Dimension Bounds** **[120]**
You attempted to APPEND FROM a SYLK file that contains one or more data items which are indicated to be outside the specified bounds of the file.

**Invalid SYLK File Format.** **[121]**
You attempted to APPEND FROM a SYLK file that either was not created by MultiPlan or has somehow been corrupted.

**Invalid SYLK File Header.** **[119]**
You attempted to APPEND FROM a SYLK file that has an invalid file header.

**^--- Keyword not found**
There is an unrecognized word in Config.db.

**Label contents must be character type.**
You entered a memo or logical field for the Label contents option in CREATE/MODIFY LABEL. Only character, numeric, and date fields are allowed.

**Label file invalid.** **[54]**
You ran or modified a LABEL file (.lbl) not created with the CREATE/MODIFY LABEL commands.

**Line exceeds maximum of 254 characters.** **[18]**
You specified a command that exceeds 254 characters in The Assistant.

**Master catalog is empty.** **[149]**
You used SET CATALOG TO ? when no catalogs have been created in the current drive or directory. Or, you have removed all catalogs from the current drive or directory.

**Maximum number of fields already reached.**
In CREATE or MODIFY STRUCTURE, you attempted to add a field to a database file that already has the maximum of 128 fields.

**Maximum path length exceeded.** **[146]**
Using SET PATH TO, you defined a path that is more than 60 characters long.

**Maximum print width of 250 characters exceeded.**
In CREATE/MODIFY LABEL, this message appears when the label width times the number of labels across is greater than 250 characters. Check the dimensions you specified for Label Width and Labels Across a Page on the Options Menu.

**Maximum record length exceeded.**
In CREATE or MODIFY STRUCTURE, you attempted to define a field width that causes the record length to exceed 4,000 bytes.

In CREATE FROM, the total of all field sizes exceeds 4,000 bytes.

**Memo and logical fields are not allowed here.**
You can create labels with character, numeric, and date type fields only.

**Memory Variable file is invalid.** **[55]**
You attempted to RESTORE FROM a file that is not a valid memory (.mem) file.

**Message file error.**
The Dbase.msg file is either damaged or missing.

**Mismatched DO WHILE and ENDDO.** **[96]**
dBASE III PLUS encountered an extra ENDDO in a command file.

**Network server busy.** **[148]**
The network server is busy.

**No database is in USE. Enter file name:**
You specified a command that requires a database file to be open in the current work area, and none are open.

**No DBF files are present.**
You tried to CREATE a VIEW when there were no database files on the disk or directory.

**No fields available.**
You tried to CREATE or MODIFY a QUERY without defining a fields list first.

**No fields of the requested type are present.**
In using one of the dBASE III PLUS menu-driven commands, you have requested a fields pull-down menu by selecting a menu option or pressing **F10**, and there are no fields of the required type either in the file structure or in the SET FIELDS list.

**No fields to process.** **[47]**
You attempted an operation such as BROWSE or EDIT with a database file that has no fields in the SET FIELDS list.

**No fields were found to copy.** **[138]**
You tried to COPY fields but did not define a fields list first.

**No files available to form this relation.**
In CREATE/MODIFY VIEW, all the files you selected while setting up the view are already in the Relation chain, and you tried to create another relation with the last file in the chain.

**No files of the requested type are cataloged.**
While using The Assistant and SET, with a catalog in use, you selected an item that needs a file type that is not in the catalog.

**No files of the requested type are present.**
While using The Assistant and SET, you selected an item that needs a specific file type that is not on the disk.

**No find.**
You attempted to execute a FIND or SEEK command with a key that does not match any entry in the indexed file.

**No format files are present in the current directory.**
In CREATE/MODIFY VIEW, you selected the Format option when there are no format files on the disk.

**No PARAMETER statement found.** **[93]**
You attempted to DO a command file or procedure WITH a parameter when there was no PARAMETER command as the first line in the command file or procedure.

**No records pass through filter.**
There are no records that meet the specified query condition.

**No room for heading.**
The heading exceeds the maximum width allowed in CREATE/MODIFY REPORT.

**Not a character expression.** **[45]**
You attempted to use an expression that is not a character type where a character expression is required.

**Not a dBASE database.** **[15]**
You attempted to USE a file that either does not conform to the dBASE III PLUS format or has a corrupted header record.

**Not a Logical expression.** **[37]**
You attempted to use an expression that is not a logical type where a logical expression is required, such as in a FOR/WHILE clause.

**Not a numeric expression.** **[27]**
You attempted to use an expression that is not a numeric type where a numeric expression is required, such as in SUM, AVERAGE, or DISPLAY NEXT.

**Not a valid PFS file.** **[140]**
You tried to IMPORT a non-PFS file or a damaged PFS file.

**Not a valid QUERY file.** **[134]**
You attempted to SET FILTER TO an invalid query file. The query file has probably been corrupted and should be re-created using CREATE QUERY.

**Not a valid VIEW file.** **[127]**
You attempted to SET VIEW TO or MODIFY VIEW using an invalid view file. The view file has probably been corrupted and should be re-created using CREATE VIEW FROM ENVIRONMENT or CREATE VIEW. The database and other files in the view are still intact.

**Not enough disk space for INDEX.**
There is not enough space on the disk to store the index file you are trying to create. You need to delete files to make room for the index file, or cancel the index operation and specify another drive on which to store files.

**Not enough disk space for SORT.**
There is not enough space on the disk to SORT a database. There must be enough space on the disk to hold two files the same size as the source database file, because SORT uses a temporary file as well as a target file.

**Not enough records to sort.**
You attempted to SORT a database file containing fewer than two records.

**** Not Found ****
You specified a search string that can not be found while using the FIND menu option in BROWSE.

**Not readable**
You used the DIR command and dBASE III PLUS was unable to read the file.

**Not suspended.** **[101]**
You issued the RESUME command when command file execution was not SUSPENDed.

**Numeric overflow (data was lost)** **[39]**
Numeric overflow occurs when a mathematical operation results in a number too large to be stored in the field or variable in which it is placed. The two common conditions that cause this are explained below under the headings REPLACE and TOTAL.

**REPLACE**

You attempted to REPLACE a numeric field with an expression that is too large.

**TOTAL**

You specified numeric fields that are too small to accommodate the totals resulting from TOTALing the database.

**Operation with Logical field invalid** **[90]**
You attempted to SORT or INDEX on a logical field.

**Operation with Memo field invalid** **[34]**
You attempted either to SORT or INDEX on a memo field, or to assign a memo field to a variable.

**Out of memory variable memory** **[21]**
You exceeded the 6,000-byte limit of memory space allocated for memory variables. You can extend this limit by changing the MVARSIZ parameter in the Config.db file.

**Out of memory variable slots** **[22]**
You attempted to initialize a memory variable with the maximum of 256 memory variables already active.

**^--- Out of range**
In the Config.db file, you specified a parameter that is not within the allowable range. See Chapter 4 for the range of allowable values.

**Page width too small or margins too large.**
In CREATE/MODIFY REPORT, the sum of the Left and Right margins is greater than or equal to the specified Page Width.

**PFS does not allow row numbers higher than 20.**
You tried to export a database file with a format file containing a row coordinate greater than 20.

**Position is off the screen.** **[30]**
You attempted to execute a SAY/GET command at a column coordinate outside the range of 0 to 79 or a row coordinate outside the range of 0 to 24.

**Printer is either not connected or turned off.** **[126]**
You attempted to SET PRINTER TO an operating system device that does not have a printer installed.

**Printer not ready. Retry? (Y/N)**
You attempted to print when the printer was not connected, not on line, or not turned on. You are given the opportunity to rectify the problem and retry printing.

**Query not valid for this environment.** **[143]**
The query file you specified does not match the database file in USE. This message appears when using SET FILTER TO FILE or CREATE/MODIFY QUERY.

**Record is in use by another.** **[109]**
You attempted to position the record pointer to a record that cannot be accessed because it is locked.

**Record is not in index** **[20]**
You attempted to position the record pointer with GO/GOTO, or any command using the RECORD phrase, to a record that is not in the index file.

**Record is not inserted**
You aborted an INSERT command without adding the record.

**Record is out of range** **[5]**
You attempted to position the record pointer to a record number greater than the number of records in the database file.

**Records do not balance (program error)**
There is something wrong with the dBASE III PLUS system. Make a new working copy of dBASE III PLUS from your master disk. If you still get the error, contact Ashton-Tate Software Support.

**Relation record is in use by others.** **[142]**
The active database file has a SET RELATION TO another file, and the record relating to the one you wish to LOCK is in use by someone else.

**Report file invalid.**
You attempted to run a REPORT FORM (.frm) file that was not created by CREATE or MODIFY REPORT.

**Report is too narrow for defined columns.**
In CREATE/MODIFY REPORT, you changed the Left Margin to a value that will not accommodate the columns already defined.

**Structure invalid.** **[33]**
You attempted to execute the CREATE FROM command when the structure defined in the structure-extended file is not valid.

**Syntax error.** **[10]**
You attempted to execute a command that does not have the proper syntactical construction. This is often caused by a misspelled word.

**Syntax error in contents expression.**
You made an error in the field contents of a LABEL.

**Syntax error in field expression.**
You made an error in the field contents of a REPORT.

**Syntax error in group expression.**
You made an error in the group expression of a REPORT.

**Syntax error in sub-group expression.**
You made an error in the subgroup expression of a REPORT.

**Table is full** **[105]**
You attempted to LOAD more than the maximum of sixteen files.

**The HELP file is not available**

You asked for HELP while in The Assistant, and the Assist.hlp file is not on the same disk or directory as DBASE.OVL.

**The query table is full.**

In CREATE/MODIFY QUERY, you pressed **Ctrl**-N to insert a line and add another condition when the maximum of seven are already defined. You can only have seven conditions on the query table.

**There are no files of the type requested in this drive or catalog. [53]**

You attempted to use a catalog query clause when there were no files of the type requested in the catalog.

**This field must contain a valid character expression.**

In the Constant/Expression portion of a condition having a character type field, you must enter a valid character expression. Check that you have spelled all field names correctly, enclosed character strings in quotes, and used character operators plus ( + ) and minus ( − ) correctly.

**This field must contain a valid date expression.**

In the Constant/Expression portion of a condition having a date type field, you must enter a valid date expression. Check that you have spelled all field names correctly.

**This field must contain a valid dBASE expression.**

In CREATE/MODIFY QUERY you entered an invalid expression for the Constant/Expression option in the Set Filter pull-down menu. In CREATE/MODIFY REPORT, you entered an invalid expression in either the Groups or the Columns pull-down menu. In any case, check the spelling of field names and the validity of the operators used in the expression.

**This field must contain a valid numeric expression.**

In the Constant/Expression portion of a condition having a numeric type field, you must enter a valid numeric expression. Check that you have spelled all field names correctly, and have used numeric operators correctly.

**This must be a valid dBASE expression.**

You attempted to enter an invalid expression on the current Label contents line in CREATE/MODIFY LABEL. The Label contents must be either a single character expression or a list of character expressions.

**This string must not contain any semicolons.**
In CREATE/MODIFY REPORT, you entered one or more semicolons for either the Title option in the Options pull-down menu or the Heading option in the Columns pull-down menu. Semicolons are not allowed in these cases. To obtain more than one Title or Heading line, use the four lines provided in the entry areas for these options.

**This view is not modifiable in full-screen view.**
You CREATEd a VIEW FROM ENVIRONMENT that cannot be changed using CREATE/MODIFY VIEW. Examples are views with disconnected relation chains, multiple filters, or multiple format files.

**Too many @ commands on one page of format file.**
You tried to EXPORT a database file with a format file that has too many @ commands on one page. You can have up to 200 lines on a page.

**Too many characters in REPORT**
You specified REPORT FORM parameters in excess of the maximum of 1,440 bytes.

**Too many files are open.** **[6]**
You attempted to open more than 15 files. This condition can also be caused when the DOS configuration file, Config.sys, which contains the clause FILES = 20, is not in the root directory of the boot disk.

**Too many indices.** **[28]**
You attempted to specify more than seven index files with either the USE or the SET INDEX TO commands.

**Too many merge steps**
There is a problem with the dBASE III PLUS system. Make a new working copy of dBASE III PLUS from your master disk. If you still get the error, contact Ashton-Tate Software Support.

**Too many pages in format file.**
You tried to EXPORT a database file with a format file that has too many pages. The limit is 32 pages per format file.

**Too many sort key fields.**
You attempted to SORT ON more than ten fields.

**Total label width exceeds maximum allowable size.**
You attempted to run a LABEL FORM in which the label width is greater than 250 columns.

**^--- Truncated**
You created a command line in Config.db that is too long and was therefore truncated.

**Two files must be in use in order to set a relation**
In CREATE/MODIFY VIEW, you attempted to use the Relate pull-down menu when only one database file was selected in Set Up.

**Unable to load COMMAND.COM.** **[92]**
Command.com was not on drive A or the drive specified by the DOS SET COMSPEC command when you:

- Issued the RUN/! command
- Executed MODIFY COMMAND or entered a memo field when an external word processor was assigned to TEDIT or WP in Config.db.

**Unable to LOCK**
In full-screen mode, you attempted to lock a record using **Ctrl**-O that was already locked by another user.

**Unable to SKIP**
While using a full-screen command such as EDIT, you attempted to access a record that is locked by another user.

**Unassigned file no.** **[2]**
There is a problem with the dBASE III PLUS system. Make a new working copy of dBASE III PLUS from your master disk. If you still get the error, contact Ashton-Tate Software Support.

**Unauthorized duplicate.**
You are using an unauthorized copy of the software.

**Unbalanced parenthesis.** **[8]**
You defined an expression with an unequal number of opposing parentheses. In CREATE/MODIFY QUERY, you have Added or Removed a parenthesis using the Nest pull-down menu, and you neglected to Add or Remove the corresponding parenthesis.

**Unknown function key.** **[104]**
In SET FUNCTION, you specified a function key that does not match the name or number of an existing function key.

**Unknown SCEDIT( ) return code:**
There is a problem with the dBASE III PLUS system. Make a new working copy of dBASE III PLUS from your master disk. If you still get the error, contact Ashton-Tate Software Support.

***** Unrecognized command verb.** **[16]**
You specified a word at the beginning of a command line that is not a valid dBASE III PLUS command.

**Unrecognized phrase/keyword in command.** **[36]**
You specified a phrase beginning with an invalid keyword.

**Unsupported path given.** **[136]**
You attempted to CREATE a database file using the ..\ as part of the path name. Give the complete name of the path.

**Unterminated string.** **[35]**
You specified a string expression without a terminating delimiter. The valid delimiters are single quotes, double quotes, and square brackets.

**Valid only in programs.** **[95]**
You issued one of the following commands at the dot prompt: CASE, DO CASE, DO WHILE, ELSE, ENDDO, IF, LOOP, OTHERWISE, SUSPEND, or TEXT. These commands are intended for dBASE III PLUS command files only.

**Variable not found.** **[12]**
You specified a non-existent memory variable or a variable name that is not a field in the currently selected work area or the work area specified by an ALIAS.

**** WARNING ** Data will probably be lost. Confirm (Y/N)**
You selected the abort option at a "disk full" message.

**Warning — label body is empty.**
This warning appears if you try to SAVE a label without defining its contents.

**** WARNING ** Report form empty.**
You attempted to exit CREATE REPORT when no fields were defined on the report.

**Wrong number of parameters.** **[94]**
You incorrectly set the DO WITH command and the PARAMETERS command with an unequal number of parameters.

# Appendix A
## CURSOR CONTROL KEYS

The control keys used in full-screen operations are:

| Key | Alternative | Function |
|---|---|---|
| ↑ | **Ctrl**-E | Moves the cursor up one line or field. |
| ↓ | **Ctrl**-X | Moves the cursor down one line or field. **Ctrl**-X also moves the cursor to the beginning of the command line at the dot prompt. |
| ← | **Ctrl**-S<br>**Ctrl**-H | Moves the cursor one space to the left. In menu bars, it moves one choice to the left. **Ctrl**-S also stops/resumes scrolling of the screen. |
| → | **Ctrl**-D | Moves the cursor one space to right. In menu bars, it moves one choice to the right. |
| **Ctrl**-→ | **Ctrl**-B | Pans one field to the right in BROWSE. In MODIFY COMMAND, moves the cursor to the end of the line. |
| **Ctrl**-← | **Ctrl**-Z | Pans one field to the left in BROWSE. In MODIFY COMMAND, moves the cursor to the beginning of the line. |
| **Backspace** | — | Erases the character to the left of the cursor. |
| **Del** | **Ctrl**-G | Erases the character at the current cursor position. |
| **End** | **Ctrl**-F | Moves the cursor one word to the right. |
| **Esc** | **Ctrl**-Q | Exits without saving changes (saves all but the current record in APPEND and BROWSE) and returns to the dot prompt. |
| **F1** | — | Toggles cursor menu on and off. |
| **Home** | **Ctrl**-A | Moves the cursor one word to the left. |
| **Ctrl-Home** | **F10** | Turns fields or options menus on in menu-assisted full-screen commands. When using the CREATE/MODIFY SCREEN command, switches between the blackboard and the Screen Painter menu bar. |

| Key | Alternative | Function |
|---|---|---|
| **Ins** | **Ctrl**-V | Toggles Insert mode on and off. When ON, inserts the character before the cursor. When OFF, overwrites the character under the cursor. |
| **Ctrl**-KB | — | In MODIFY COMMAND and Memo fields, reformats a paragraph. (Hold down the **Ctrl** key, type the letter K, and then the letter B.) |
| **Ctrl**-KF | — | In MODIFY COMMAND, finds the first occurrence of a specified string. |
| **Ctrl**-KL | — | In MODIFY COMMAND, finds the next occurrence of the string found with **Ctrl**-KF. |
| **Ctrl**-KR | — | IN MODIFY COMMAND, reads another file into the file. |
| **Ctrl**-KW | — | In MODIFY COMMAND, writes the entire file to another file. |
| **Ctrl**-N | — | Inserts a new line or field definition. |
| **Ctrl**-P | — | Toggles the printer on and off. |
| **PgUp** | **Ctrl**-R | Moves back to the previous record, screen display, or 17-record window in BROWSE. (Window shows 10 records when Help is displayed.) |
| **Ctrl-End** | **Ctrl**-W | Zoom out. Exit and save for full- screen operations. |
| **PgDn** | **Ctrl**-C | Moves to the next record, screen display, or 17-record window in BROWSE. (Window shows 10 records when Help is displayed.) |
| **Ctrl-PgDn** | — | Zoom in. Enter and edit memo fields. In CREATE/ MODIFY REPORT or LABEL, enter editing line for text or expressions. |
| **Ctrl-PgUp** | — | Zoom out. Exit and save memo fields. In CREATE/ MODIFY REPORT or LABEL, exit editing line for text or expressions. |

## CURSOR CONTROL KEYS

| Key | Alternative | Function |
|---|---|---|
| ↵ | — | Moves the cursor to the next field or line. In APPEND, it saves and exits if it is on the first character of a blank record. In EDIT, it saves and exits if it is in the last field of the last record. In MODIFY COMMAND, it inserts a line if in Insert mode. In menus, it selects an option. |
| **Ctrl**-T | — | Erases one word to the right of the cursor position. |
| **Ctrl**-U | — | Marks a record for deletion in BROWSE or EDIT. Deletes a field definition in MODIFY REPORT or MODIFY STRUCTURE. Deletes a field, box, or line in CREATE/MODIFY SCREEN. |
| **Ctrl**-Y | — | Erases to the end of the field in APPEND or EDIT, or erases the entire line in MODIFY COMMAND, CREATE/MODIFY SCREEN, or when editing the contents of a memo field. |

# Appendix B

## BEGINNING/END OF FILE AND FILTER CONDITIONS

### Unindexed Files

For all unINDEXed files, the following conditions exist when the specified command sequence is executed:

| Command | RECNO() | EOF() | BOF() | Error* |
|---|---|---|---|---|
| GO TOP | 1 | .F. | .F. | N |
| GO TOP; SKIP -n | 1 | .F. | .T. | N |
| GO TOP; SKIP -n; SKIP m | m + 1 | .F. | .F. | N |
| GO TOP; SKIP -n; SKIP -m | 1 | .F. | .T. | Y |
| GO BOTTOM | last | .F. | .F. | N |
| GO BOTTOM; SKIP n | last + 1 | .T. | .F. | N |
| GO BOTTOM; SKIP n; SKIP -m | (last + 1) − m | .F. | .F. | N |
| GO BOTTOM; SKIP n; SKIP m | last + 1 | .T. | .F. | Y |
| LIST | last + 1 | .T. | .F. | N |
| LOCATE (found) | n | .F. | .F. | N |
| LOCATE (not found) | last + 1 | .T. | .F. | N |

* Beginning-of-file and end-of-file error messages are generated if the condition is already True before the indicated command is issued.

Note that "last" refers to the number of the last record in the database. Any attempt to move past that record yields a RECNO() equal to the total number of records plus 1. m and n refer to positive integer values.

All commands that *do not* cause repositioning of the database have no effect if EOF() is True.

Table B-1 Beginning and end-of-file conditions for unindexed files

## Indexed Files

For all INDEXed files, the following conditions exist when the specified command sequence is executed. RECNO() refers to the number of the record at the given position:

| Command | RECNO() | EOF() | BOF() | Error* |
|---|---|---|---|---|
| GO TOP | first | .F. | .F. | N |
| GO TOP; SKIP -n | first | .F. | .T. | N |
| GO TOP; SKIP -n; SKIP m | first + m | .F. | .F. | N |
| GO TOP; SKIP -n; SKIP -m | first | .F. | .T. | Y |
| GO BOTTOM | last | .F. | .F. | N |
| GO BOTTOM; SKIP n | last + 1 | .T. | .F. | N |
| GO BOTTOM; SKIP n; SKIP -m | (last + 1) − m | .F. | .F. | N |
| GO BOTTOM; SKIP n; SKIP m | last + 1 | .T. | .F. | Y |
| LIST | last + 1 | .T. | .F. | N |
| LOCATE (found) | n | .F. | .F. | N |
| LOCATE (not found) | last + 1 | .T. | .F. | N |
| FIND (found) | n | .F. | .F. | N |
| FIND (not found) | last + 1 | .T. | .F. | N |
| SEEK (found) | n | .F. | .F. | N |
| SEEK (not found) | last + 1 | .T. | .F. | N |

* Beginning-of-file and end-of-file error messages are generated if the condition is already True before the indicated command is issued.

Note that "last" refers to the position of the last record in the index. Any attempt to move past that record yields a RECNO() equal to the total number of records plus 1.

All commands that *do not* cause repositioning of the database have no effect if EOF() is True.

Table B-2 Beginning and end-of-file conditions for indexed files

## Filter Conditions

The effects of SET DELETED and SET FILTER on the records processed by dBASE commands are described in the following table.

| | SET DELETED OFF<br>NO FILTER | SET DELETED ON<br>FILTER ON |
|---|---|---|
| GO TOP | Logical first record. | Logical first active record. EOF() is True if no records are active. |
| GO BOTTOM | Logical last record. | Logical last active record. BOF() and EOF() are True if no records are active. |
| GOTO | Specified record. | Specified record. |
| SKIP | Current record +/− specified amount. BOF() and EOF() are set accordingly. | Current record +/− specified amount of active records. BOF() and EOF() are set accordingly. |
| FIND<br>SEEK<br>LOCATE | First occurrence of specified condition. EOF() is True if none are found and the LOCATE scope is ALL. | First occurrence of specified condition in an active record. EOF() is True if none are found and the LOCATE scope is ALL. |
| DELETE | Delete specified records, position on last deleted record. | Delete specified active records, position on last deleted record. |
| COPY<br>COUNT<br>EDIT<br>LIST<br>RECALL<br>REPLACE<br>SORT<br>SUM | Process all records in specified scope or until EOF() is True. | Process active records in specified scope or until EOF() is True. The current record is included (active or not) if the scope is NEXT n or REST. |
| INDEX<br>REINDEX | Include all records. | Include all records. |

NOTE: An active record is one that is not hidden by the SET FILTER condition, or is not marked for deletion.

Table B-3 Filter condition effects on dBASE commands

# Appendix C
## STRUCTURE OF DATABASE (.DBF) AND MEMO (.DBT) FILES

### Structure of a Database File

A database (.dbf) file is composed of a header, data records, and an end-of-file marker. The header contains information about the file setup, while the data record portion contains the actual data in the file that you display, add, or update.

### The Header Structure

The header structure provides information dBASE III PLUS uses to maintain the database file. The structure of the header is detailed in Tables C-1 and C-2.

| Byte | Contents | Meaning |
|---|---|---|
| 0 | 1 byte | Valid dBASE III PLUS file (03H without a memo (.dbt) file; 83H with a .dbt file) |
| 1–3 | 3 bytes | date of last update; formatted as *YY MM DD* |
| 4–7 | 32-bit number | number of records in data file |
| 8–9 | 16-bit number | number of bytes in the header |
| 10–11 | 16-bit number | number of bytes in the record |
| 12–14 | 3 bytes | reserved bytes |
| 15–27 | 13 bytes | reserved for dBASE III PLUS on a local area network |
| 28–31 | 4 bytes | reserved bytes |
| 32–n† | 32 bytes each | field descriptor array (the structure of this array is shown in Table C-2) |
| n + 1 | 1 byte | 0DH as the field terminator |

† n is the last byte in the field descriptor array. The size of the array depends on the number of fields in the database file.

Table C-1 Database file header

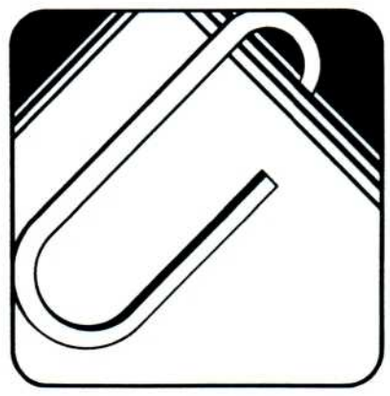
| Byte | Contents | Meaning |
|---|---|---|
| 0–10 | 11 bytes | field name in ASCII (zero-filled) |
| 11 | 1 byte | field type in ASCII (C, N, L, D, or M) |
| 12–15 | 32-bit number | field data address (address is set in memory; not useful on disk) |
| 16 | 1 byte | field length in binary |
| 17 | 1 byte | field decimal count in binary |
| 18–19 | 2 bytes | reserved for dBASE III PLUS on a local area network |
| 20 | 1 byte | work area ID |
| 21–22 | 2 bytes | reserved for dBASE III PLUS on a local area network |
| 23 | 1 byte | SET FIELDS flag |
| 24–31 | 8 bytes | reserved bytes |

Table C-2 Database field descriptor bytes

## The Structure of Database (.dbf) File Records

Following the header is the actual data in the database file. The end of the file is marked by a single byte. The rules for structuring data records are:

1. Data records are preceded by one byte, a space (20H) if the record is not deleted, an asterisk (2AH) if the record is deleted.
2. Data fields are packed into records and have no field separators or record terminators.



3. Data types are stored in ASCII format as follows:

| Data Type | Data Record Storage |
|---|---|
| Character | ASCII characters |
| Numeric | - . 0 1 2 3 4 5 6 7 8 9 |
| Logical | ? Y y N n T t F f (? when not initialized) |
| Memo | 10 digits representing a .dbt block number |
| Date | 8 digits in YYYYMMDD format, such as 19840704 for July 4, 1984 |

Table C-3 ASCII codes for data types

4. The end-of-file marker is an ASCII 26 (1AH) character.

## The Structure of Database Memo (.dbt) Files

Memo (.dbt) files consist of 512-byte blocks numbered sequentially (0, 1, 2, and so on). Each time you write to the memo field in a record of your database file, another block is added to the memo file.

The first block in the memo file, block 0, is the memo file header. Only the first four bytes in the header block are used. That's where dBASE III PLUS indicates the next available block number. The number is hexadecimal, byte one containing the least significant portion. For example, if the next available block is number 200, the first four bytes of the .dbt file are C8 00 00 00 (hexadecimal). If the next available block is number 4321, the bytes are E1 10 00 00, and so on. The following table provides an example of how memo file blocks are used.

| Block Number | .dbt Datafile |
|---|---|
| 0 | 05 00 00 00 . . . |
| 1 | first memo field saved |
| 2 | second memo field saved (exceeds 512 bytes) |
| 3 | still second memo field |
| 4 | third memo field saved |
| 5 | next available block (not yet in file) |

Table C-4 The structure of .dbt files

Each memo field of each record in the .dbf file contains the number of the block (in ASCII) where the memo field data is located. If the memo field contains no data, there is no number in the .dbf file. When you write to the memo field, the next available block is used, and its number is stored invisibly in ASCII in the memo field of that record, behind the word *memo*.

When data is changed in a memo field, the old contents remain in the .dbt file, and the entire changed contents are rewritten to the next available block. The number in the .dbf is changed to reflect the new location. This leaves the old block unused, with no pointer to it in the .dbf, and explains why .dbt files grow so large when edited repeatedly. When the file is copied from within dBASE III PLUS with USE <filename> and COPY TO <filename>, only the blocks that are pointed to get copied, thus discarding blocks no longer used and recovering the disk space.

If the data written to a memo field uses more than one 512-byte block, the next available block number is incremented by the number of blocks added. In other words, if the next available block is number 15, after 1200 bytes (which take up 3 blocks) are entered, the next available block is 18.

Each memo field terminates with two hex 1A bytes. If 520 characters are entered into a memo field, the ninth and tenth bytes of the next block will hold these terminators. The remaining 502 bytes of this block will be unused.

# Appendix D
## ASCII CODE TABLES

### Printable Characters

| Hex | Decimal | Symbol | Hex | Decimal | Symbol |
|---|---|---|---|---|---|
| 20 | 32 | SPACE | 43 | 67 | C |
| 21 | 33 | ! | 44 | 68 | D |
| 22 | 34 | " | 45 | 69 | E |
| 23 | 35 | # | 46 | 70 | F |
| 24 | 36 | $ | 47 | 71 | G |
| 25 | 37 | % | 48 | 72 | H |
| 26 | 38 | & | 49 | 73 | I |
| 27 | 39 | ' | 4A | 74 | J |
| 28 | 40 | ( | 4B | 75 | K |
| 29 | 41 | ) | 4C | 76 | L |
| 2A | 42 | * | 4D | 77 | M |
| 2B | 43 | + | 4E | 78 | N |
| 2C | 44 | , | 4F | 79 | O |
| 2D | 45 | - | 50 | 80 | P |
| 2E | 46 | . | 51 | 81 | Q |
| 2F | 47 | / | 52 | 82 | R |
| 30 | 48 | 0 | 53 | 83 | S |
| 31 | 49 | 1 | 54 | 84 | T |
| 32 | 50 | 2 | 55 | 85 | U |
| 33 | 51 | 3 | 56 | 86 | V |
| 34 | 52 | 4 | 57 | 87 | W |
| 35 | 53 | 5 | 58 | 88 | X |
| 36 | 54 | 6 | 59 | 89 | Y |
| 37 | 55 | 7 | 5A | 90 | Z |
| 38 | 56 | 8 | 5B | 91 | [ |
| 39 | 57 | 9 | 5C | 92 | \ |
| 3A | 58 | : | 5D | 93 | ] |
| 3B | 59 | ; | 5E | 94 | ^ |
| 3C | 60 | < | 5F | 95 | _ |
| 3D | 61 | = | 60 | 96 | ` |
| 3E | 62 | > | 61 | 97 | a |
| 3F | 63 | ? | 62 | 98 | b |
| 40 | 64 | @ | 63 | 99 | c |
| 41 | 65 | A | 64 | 100 | d |
| 42 | 66 | B | 65 | 101 | e |

| Hex | Decimal | Symbol | Hex | Decimal | Symbol |
|---|---|---|---|---|---|
| 66 | 102 | f | 73 | 115 | s |
| 67 | 103 | g | 74 | 116 | t |
| 68 | 104 | h | 75 | 117 | u |
| 69 | 105 | i | 76 | 118 | v |
| 6A | 106 | j | 77 | 119 | w |
| 6B | 107 | k | 78 | 120 | x |
| 6C | 108 | l | 79 | 121 | y |
| 6D | 109 | m | 80 | 122 | z |
| 6E | 110 | n | 7B | 123 | { |
| 6F | 111 | o | 7C | 124 | \| |
| 70 | 112 | p | 7D | 125 | } |
| 71 | 113 | q | 7E | 126 | ~ |
| 72 | 114 | r | 7F | 127 | DEL |

Table D-1 ASCII printable character codes

## Control Characters

| Hex | Decimal | Symbol | Code | Description |
|---|---|---|---|---|
| 00 | 0 | NUL | ^@ | Null |
| 01 | 1 | SOH | ^A | Start of Heading |
| 02 | 2 | STX | ^B | Start of Text |
| 03 | 3 | ETX | ^C | End of Text |
| 04 | 4 | EOT | ^D | End of Transmission |
| 05 | 5 | ENQ | ^E | Enquiry |
| 06 | 6 | ACK | ^F | Acknowledge |
| 07 | 7 | BEL | ^G | Bell |
| 08 | 8 | BS | ^H | Backspace |
| 09 | 9 | SH | ^I | Horizontal Tabulation |
| 0A | 10 | LF | ^J | Line Feed |
| 0B | 11 | VT | ^K | Vertical Tabulation |
| 0C | 12 | FF | ^L | Form Feed |
| 0D | 13 | CR | ^M | Carriage Return |
| 0E | 14 | SO | ^N | Shift Out |
| 0F | 15 | SI | ^O | Shift In |
| 10 | 16 | DLE | ^P | Data Link Escape |
| 11 | 17 | DC1 | ^Q | Device Control 1 |
| 12 | 18 | DC2 | ^R | Device Control 2 |
| 13 | 19 | DC3 | ^S | Device Control 3 |
| 14 | 20 | DC4 | ^T | Device Control 4 |
| 15 | 21 | NAK | ^U | Negative Acknowledge |
| 16 | 22 | SYN | ^V | Synchronous Idle |
| 17 | 23 | ETB | ^W | End of Transmission Block |
| 18 | 24 | CAN | ^X | Cancel |
| 19 | 25 | EM | ^Y | End of Medium |
| 1A | 26 | SUB | ^Z | Substitute |
| 1B | 27 | ESC | ^[ | Escape |
| 1C | 28 | FS | ^\ | File Separator |
| 1D | 29 | GS | ^] | Group Separator |
| 1E | 30 | RS | ^^ | Record Separator |
| 1F | 31 | US | ^_ | Unit Separator |
| 7F | 127 | DEL | DEL | Delete |

Table D-2 ASCII control character codes

# Glossary

This section defines terms used with dBASE III PLUS which may also have special meaning when used to describe dBASE commands and functions.

**?** Catalog query clause. Activates a pull-down menu list in the open catalog that displays files of a special type, such as .frm files with the command CREATE REPORT ?.

Command used to display immediate results of expressions or the contents of fields, for example, ? SQRT(16) or ? Lastname,firstname.

**Abandon** An option which terminates a dBASE III PLUS command operation. Any changes which have been made are not saved.

**Action Line** A line near the bottom of a menu that is used to display information about the application in progress.

**Active File** The database file in USE in the currently selected work area.

**Alias** An alternative database filename by which its work area can be referenced. An alias may be up to 10 characters in length and can contain letters, numbers, and underscore characters. It must begin with a letter and may not contain embedded blank spaces. Do not use the single letters A through J or M as file or alias names, because these are the default alias names for the 10 work areas and memory variables.

**Application** A customized program written in the dBASE programming language that performs tasks required by a specific situation. The checkbook management system used in *Programming with dBASE III PLUS* is an application.

**Applications Generator** A special program on the Applications Generator disk that helps users quickly create or modify database files or simple programming applications.

**Ascending Order** An order from the lowest to the highest value (for example, A to Z and 0 to 9). In dBASE III PLUS, ascending order for character data is determined by the ASCII value. See Appendix D for a list of ASCII character codes.

**ASCII** An acronym for American Standard Code for Information Interchange. It is a seven-bit code that defines 128 standard characters, including control characters, letters, numbers, and symbols. An extra 128 characters are part of the extended ASCII set. The seven-bit ASCII character codes are listed in Appendix D.

**Assembly Language** A programming language that directly controls the machine code of a specific computer's microprocessor.

**Attribute** A descriptive characteristic. In this manual, attributes refer to the characteristics of the video display, including intensity, inverse video, and underline.

**Background** The area on the monitor upon which characters appear.

**Back up** The process of copying all or some of the files on a disk to another disk as insurance against the possible failure or loss of the first disk.

**Backup File** A duplicate file copied on a separate disk and kept in a safe location as insurance against loss of the original file.

**Batch** A sequence of commands that perform a particular task and can be executed with a single overall command. Also, a group of records that may be considered as a single unit for the purposes of processing.

**Blackboard** The screen area used to design screen entry forms in the Screen Painter.

**Beginning-of-File (BOF)** A location before the first physical record of a file.

**Block** A unit of data as recorded on a disk. In MS-DOS, disks are divided into sectors, each of which contains a block of 512 bytes. The COPY FILE command transfers data in blocks only, meaning that the size of the file created is rounded up to the next 512-byte unit.

## GLOSSARY

**Branching** — An unconditional change in program flow whereby commands in another program, usually a subprogram, are performed. When the subprogram commands are finished, program control returns to the calling program.

**Calling Program** — A program module which contains instructions to branch to other programs, known as subroutines.

**Catalog** — A special class of database file used to track dBASE III PLUS files that are associated with a database file. Each associated file becomes one record in the catalog. A catalog keeps records on all files except command, text, and memory files. A catalog also allows selection of files from pull-down menus, rather than by specifying a filename.

**Character** — A keyboard symbol. Characters include letters, numbers, blank spaces, and special symbols.

**Character String** — A sequence of letters, numbers, or symbols. The sequence *1234 Main Street, Anytown,* is a character string. In this example, blanks are included as part of the character string. Character strings are normally enclosed in quotation marks or some other type of delimiter.

**Chronological** — In date order. Dates do not appear in chronological order when converted to character strings, unless put in the format of 'YYYYMMDD'.

**Column** — A number indicating the horizontal position on either the video monitor or the printer. Column values increase from the left, with the smallest value being zero. Also used to explain a field as a column in a rectangular table of rows and columns. Column is also used to refer to a REPORT FORM column, that is, the content that falls under a heading.

**Command** — An instruction to the computer to carry out a specific operation. A verb that begins an instruction to dBASE III PLUS and appears as the first word in a command line.

**Command File** — A computer program containing dBASE III PLUS commands and saved as a disk file with a .prg extension.

**Concatenation** — The joining of two or more character strings to form a single character string. In dBASE III PLUS, character strings may be concatenated with the plus ( + ) or minus ( − ) sign.

**Condition** — An expression that evaluates to a logical value of .T. or .F.

**Conditional Replacement** — A technique for changing the contents of records in a database file where the replacement of data depends on a specified condition being met.

**Constant** — A data item that does not change during an operation. A constant is also referred to as a literal.

**Control Key** — A key that assigns another function to certain keys on the computer keyboard. Like the shift key, the control key is held down while another key is pressed.

**Control Structure** — A way to alter the normal step-by-step flow of a program. In dBASE, control structures are DO (branching), DO WHILE...ENDDO (looping), and IF...ENDIF and DO CASE...ENDCASE (conditions).

**Coordinate** — A number, referring to a specific screen row and column position, used as a reference point for displaying information.

**Counter** — A display that shows the number of items currently processed by a command.

**Cstring** — An abbreviation for character string.

**Current Option** — The highlighted menu option.

**Current Record** — The record at which the record pointer is positioned. The record in use.

**Cursor** — The visible marker used to indicate the current position on the screen.

## GLOSSARY

**Cursor Control Menu** — The menu of navigation keys. Pressing **F1** toggles this menu on and off in full-screen operations such as EDIT and BROWSE.

**dBASE** — The programming language built into dBASE III PLUS.

**DBMS** — An acronym for DataBase Management System.

**Data Disk** — A disk containing the database, index, format, and other files used by dBASE III PLUS.

**Data Type** — One of the five types of data that can be entered in a database file: character (C), numeric (N), logical (L), date (D), or memo (M). Memory variables also have data types, but no equivalent to memo (M).

**Database** — A file or group of files containing records of related data. The database may also include any ancillary files to the database files such as index, report, and query files.

**Database Integrity** — The accuracy of the data in a database.

**Debug** — To find and correct errors in a software program.

**Decrement** — To decrease a value, or the amount by which a value is decreased.

**Default** — The preset or automatic value of an option in the absence of a different specification. Defaults in dBASE III PLUS environments are usually changed with a SET command. Command defaults are usually changed by a scope or conditional clause.

**Definition** — A filename in a list of input files that dBCODE and dBLINKER use in processing.

**Delimiters** Characters used to enclose character data items. Character data is normally delimited with single or double quotation marks (for example, STORE 'Hello' TO X). Square brackets are also valid delimiters in dBASE III PLUS. Commas are used to separate data items in a list (for example, DISPLAY Name,Address,City,State), or to separate fields in a text file. When used to separate fields in a text file, the comma is properly called a field separator.

**Descending Order** The order from the highest to the lowest value (for example, Z to A and 9 to 0). In dBASE III PLUS, descending order for character data is determined by reverse ASCII order. See Appendix D for a list of ASCII character codes.

**Device** Any part of the computer that is capable of sending or receiving information. The keyboard is an example of an input device; the video monitor and printer are output devices.

**Directory** A catalog of files on a disk.

**DOS** An acronym for Disk Operating System. Often used to refer to the PC-DOS or MS-DOS operating system.

**Dot Prompt** The dBASE III PLUS prompt, represented by a dot (.) on the screen (unless changed in the Config.db file). This is also known as the interactive mode, as opposed to the full-screen mode. Editing capabilities such as inserting, deleting, and overwriting are available at the dot prompt.

**Drive** Indicates the disk drive identifier (for example, A, B, or C). When used as part of a filename, the disk drive identifier must be followed by a colon (for example, A:).

**Encrypt** To make program code or a database file unreadable by others so that it cannot be modified or otherwise tampered with. The dBCODE portion of RunTime + encrypts dBASE program files.

**End-of-File (EOF)** A location after the last physical record.

## GLOSSARY

**Enhanced** — An attribute of the monitor display used by some dBASE III PLUS commands. This appears on a monochrome monitor as black foreground on a white background. See Standard.

**Enter** — Used in two contexts: (1) to type in a command or data in response to a prompt message, and (2) a synonym for the RETURN key.

**Enumerated Value** — One of a fixed number of possible values for a menu option. For example, one of the five possible field types that can be assigned while creating a database file, or the ON/OFF values in the SET Menu.

**Environment** — See Working Environment.

**Error Trapping** — Refers to routines set up in programs to check for and correct potential user errors.

**Esc** — See Escape Key.

**Escape Code** — A symbol used to instruct a printer to print in a special mode, such as expanded print mode, by combining of the ASCII code for **Esc** (Decimal Code 27), with one or more other ASCII codes.

**Escape Key** — The key used in dBASE III PLUS to abandon program or command execution.

**Exp** — See Expression.

**Expression** — A combination of constants, variables, and operators that produces a result of a single data type, that is, numeric, character, logical, or date. The type of an expression is defined by the type of result that it returns. Dates may enter into expressions with numbers (addition and subtraction) to form date expressions. One date may be subtracted from another to form a numeric expression. The letters C, D, L, or N may be used after the abbreviation to indicate the data type of the expression. For example, <expC> indicates a character expression.

**Expression List** An expression list consists of one or more expressions. If more than one expression is present, they must be separated by commas. The expressions do not have to be the same data type.

**Extension** A period and up to three characters added to the end of a filename that can be used to identify the type of file. In the sample file Clients.dbf, the extension is .dbf, which stands for "database file." The extension is also called a file identifier.

**Field** An individual data item in a record within a database file.

**Field List** A listing of field names separated by commas.

**Field Name** The name assigned to a field. A field name may have up to 10 characters consisting of letters, numbers, and underscores. It must begin with a letter and cannot contain embedded blank spaces.

**Field Type** Indicates the type of data that may be stored within a field. Field types are character (C), date (D), logical (L), memo (M), and numeric (N).

**Field Width** The number of character spaces assigned to a field.

**File** A collection of related information. When used within a command, FILE followed by a filename indicates that the command is to apply to the entire file (for example, DELETE FILE Garbage.dbf).

**Filename** The name of a disk file, which may consist of up to eight characters and may be followed by an optional period and a three-letter extension. It must begin with a letter and cannot contain an embedded blank space. Do not use the single letters A through J or M as filenames.

**File Type** An aspect of a file describing its structure or contents, or the function it performs with a program. File types are often identified by the three-character extension following the filename and a period. For example, in dBASE III PLUS, database files are assigned an extension of .dbf to indicate they hold data.

## GLOSSARY

**Filter**
An expression that limits the data used in commands that require a database (.dbf) file to be in USE. It does not alter the size of the file.

**FOR <condition>**
An optional clause in many commands which indicates that the command is to apply to every record in the database for which the scope of the condition is True.

**Foreground**
The characters that appear on the monitor against the background. See Background.

**Form Feed**
An action represented by ASCII code 12 that causes the paper in a printer to advance to the top of the next page. The EJECT command causes a form feed.

**Full-Screen Command**
A dBASE III PLUS command that occupies the entire monitor screen with menu, help, or message text while the command is in progress. EDIT and The Assistant (ASSIST) are full-screen commands.

**Function**
Part of the dBASE command language. A function accepts arguments of a particular type and returns data of a specified type. A function is an expression the type of which is defined by its result.

**Hard Carriage Return**
ASCII code 13. It is equivalent to a carriage return and line feed.

**Header**
Information at the beginning of a database file that allows dBASE III PLUS to maintain the file, such as the file type and the number of fields and records (see Appendix C).

**Hidden Variable**
A PUBLIC variable the value of which is hidden from a subprogram when a PRIVATE variable is created with the same name. When program flow returns to the calling program, the original PUBLIC variable is reinstated.

**Highlight**
An inverse video bar that surrounds the text of the current option on a menu.

**History** — A feature that retains command lines typed at the dot prompt or executed in a program file and allows them to be recalled and reused or (if entered at the dot prompt) edited. The command lines are stored in a memory buffer called the history buffer, which stores the last 20 lines unless its size is modified in Config.db or with SET HISTORY TO.

**Hook** — Files named in a DO <filename> statement but not provided to dBLINKER for processing. RunTime+ does not alter these undefined references, allowing easy modification of applications.

**Horizontal Scrolling** — A way to allow input of data into a @...GET field or entry area of a menu when the data is longer than the blank. This is done with the S<n> template function for the @ command, and occurs automatically in menus.

**Housekeeping** — Those functions necessary to the operation of a computer but not directly related to the program in progress. Examples of housekeeping include opening and closing files, clearing memory, and verification functions.

**I/O** — Abbreviation for input/output. A disk drive is an example of an input/output device.

**Increment** — To increase a value, or the amount by which a value is increased.

**Inactive Option** — An option that cannot be selected from a menu because circumstances make it inappropriate.

**Index File** — A special-purpose file that relates a key (an item of particular interest to the user) to a database record number. Index keys are in ascending order.

**Initialize** — To set up memory variables by STOREing data in a named variable.

**Input** — The transfer of data into the CPU or RAM of the computer, such as input from the keyboard.

**Instruction** — A command or program step telling the computer exactly what to do for a particular operation.

**Integer** A number with no fractional part; a whole number.

**Interpreter** The part of dBASE III PLUS which parses commands and performs them.

**Interrupt** To prematurely stop a program or operation in progress.

**Install** To transfer a functional copy of dBASE III PLUS to a disk other than the original system disk. Installing is automated by use of the INSTALL batch program on System Disk #1.

**Interactive** Requiring the participation of the user. Interactive commands or programs issue prompts and accept input.

**Inverse Video** Displaying text on the monitor by using dark letters against a light background. Also referred to as reverse video.

**Key** A field or expression value by which a database file is INDEXed or SORTed.

**Key Expression** An expression used to INDEX or SORT a database file.

**Key Field** A field which is used alone, or as part of a key expression, to INDEX or SORT a database file.

**Leading Blanks** The blank spaces preceding the first non-blank character in a character variable.

**Linking** Using dBLINKER to join encrypted program modules (except procedure files) to create a finished, complete program. Also used to describe the process of setting a relation between database files.

**Literal** An explicit representation to be taken at face value; a word, number, or symbol that describes or defines itself and not something else that it might represent. Literals are never delimited.

**Logical Operator** One of the operators .AND., .OR., or .NOT. used to relate logical expressions.

**Logical Order** The order of records based on an index file.

**Log-in Code** — A code used to identify a particular user to the system and allow entry.

**Looping** — The repeated execution of a series of commands while a given condition is True.

**Macro** — A single word or symbol that substitutes for a group of commands or an entire subroutine. The ampersand (&) is the macro function. It substitutes the contents of a memory variable in those cases where dBASE III PLUS requires a literal value.

**Main Program** — The highest-level calling program in a dBASE programming application.

**Master Catalog** — A special catalog file, automatically created and maintained in the directory where dBASE III PLUS is installed, that tracks all other catalog files on a disk.

**Memory File** — A disk file used to permanently store memory variables.

**Memory Variable** — An identifiable data item that is not part of a database file but is stored in the computer's memory for temporary use.

**Memvar** — An abbreviation for memory variable. It normally indicates that the name of a memory variable is to be used.

**Memvar List** — Consists of one or more memory variable names. If more than one memory variable is present, the names must be separated by commas.

**Menu** — A list of options.

**Menu-assisted** — Used to describe commands that have an optional menu.

**Menu-driven** — The process that allows users to select a course of action based upon one or more user-selectable options. HELP and The Assistant are examples of menu-driven commands.

**Monitor** — The video display device or video screen.

**Multi-user** A system that allows two or more users to share the resources of a computer, its peripherals, and in some cases, its data.

**n** Indicates a number. In most cases, a numeric expression (such as cost * percentage) may be used as well as a literal number (such as 5).

**Nest** In programming, putting a control structure, such as a DO WHILE...ENDDO or IF...ENDIF construction, within another control structure. In query files, to put parentheses around conditions to redefine the order of evaluation.

**Network** A group of computer work stations linked to a common processor or file server.

**Numeric Fields** Fields containing numbers that may be directly used in mathematical calculations.

**Operator** A symbol used to indicate how two items are to be related. For example, the plus sign ( + ) is the arithmetic operator for addition and the string operator for concatenation.

**Output** The transfer of data from the computer to a peripheral device such as a printer, disk drive, or monitor.

**Pan** To move off-screen information into view in BROWSE and other commands.

**Parameter**

1. A value supplied to a command, function, or subroutine.
2. The adjustable processing characteristics of dBASE III PLUS that are controlled by the SET command. For example, the printer status (ON or OFF) is a processing parameter.
3. PARAMETERS is also a command that allows a subroutine to access field names, memory variables, and expressions passed to it with a DO command.

**Parameter Passing** Giving a set of variables to another program for use in that program. See Parameter.

**Parse** — The initial test dBASE III PLUS does on a command line to verify the correct syntax of the command.

**Password** — An identifying code word that allows a user access to a system or file.

**Path** — Refers to the specified search pattern to locate files not on the current directory.

**Prefix** — A (possibly null) text string affixed before a filename. This may be used to specify a drive and/or subdirectory, such as A: < filename > .

**Print** — Indicates output to the printer within a command. TO PRINT is an optional part of certain commands that turns the printer on only while the command is active. The printer can also be turned on and off by means of the SET PRINT and SET DEVICE commands.

**Procedure** — A special type of dBASE III PLUS program or subroutine. A procedure file may contain up to 32 routines (called procedures), each beginning with the word PROCEDURE.

**Procedure File** — A special type of dBASE III PLUS command file. It may contain up to 32 procedures. Its function is to combine several related command files into one file.

**Program** — A set of instructions or steps designed to have the computer perform a specific sequence of actions; an application developed in the dBASE programming language.

**Programming** — A way to set up commands in special files without having to retype the commands each time they are needed.

**Programming Language** — A special set of syntax rules and commands that tells the computer what to do and in what sequence.

**Prompt** — A symbol, word, or phrase that is used to indicate the computer's readiness to accept input from the user and the input expected.

## GLOSSARY

**Pseudo-Code** — A way of writing programs that incorporates both English and actual program commands as a method of clarifying program structure during the design stage of a programming project.

**Pull-Down Menu** — A menu that displays selection options for its related heading displayed on the menu bar.

**Query** — A specific request for data or instructions.

**Query Clause** — See the entry under "?".

**Record** — A group of related fields of information treated as a unit. Also, one complete horizontal row in a rectangular table of rows and columns.

**Record Count** — The total number of records in a database file.

**Record Number** — A unique identifying number assigned by dBASE III PLUS to each record in a database.

**Record Pointer** — A number used by dBASE III PLUS to identify the current database record.

**Reference** — The file named in a DO <filename> statement.

**Relation** — A connection established between multiple database files by using a key field or expression as a common reference point.

**Relational Database System** — A system that allows data to be accessed based on relationships which are set up among several database files.

**Relative Addressing** — Using row and column positions that relate to the previous position on the screen or printer without using specific coordinate numbers.

**Routine** — A set of commands that performs a particular operation.

**Row** — A number indicating the vertical position on either the video monitor or the printed page. Rows are numbered from top to bottom; the top line on both devices is row 0. When referring to the printer, the word *line* is used synonymously with *row*.

**RunTime +** The dBASE III PLUS system for creating commercial applications. RunTime + includes dBCODE for compiling command and procedure files, and dBLINKER for joining these files into a single file.

**Run-Time Error** A mistake in programming logic that generally doesn't appear until the program is run.

**Scope** An optional part of many commands that specifies the amount of the database file to which the command applies. Each command which has the SCOPE option has a default of either the current database record or all database records. The scope may be: RECORD N — a single record; NEXT N — n records beginning with the current record; ALL — all of the records in the database file; or REST — from the current record to the end of the file.

**Scoreboard** Information that is pertinent to data entry, such as insert and overwrite status, or the deletion status of records. Scoreboard information is displayed either on the status bar in full-screen commands, or at the top of the screen when working from the dot prompt with SET STATUS OFF.

**Screen** Any display seen on the monitor. Also, a particular image that may be referenced.

**Screen Form** A customized screen entry form used to receive user input. Screen forms generally replace those supplied by default in the full-screen commands, such as APPEND or EDIT.

**Screen Painter** A feature in dBASE III PLUS that allows users to create customized screen forms easily without knowing specific row and column coordinates.

**Sector** In MS-DOS, disks are divided into logical areas called sectors. Each sector contains a block of 512 bytes. The COPY FILE command transfers data in sectors rather than by individual bytes. This means that the file created will only contain complete 512-byte blocks, even if the last block is only partially filled.

**Sequential Access** A method by which the database is searched in consecutive order until the desired record is located.

**Setup Area** — The part of the main program that determines the working environment of the program.

**Single-user** — That version of dBASE III PLUS configured for use on stand- alone microcomputer systems.

**Skeleton** — The mask or template used to define filenames and memory variable names that have common elements. A skeleton makes use of the ? and the * to "wildcard" parts of these names with common elements. For example, DIR *.ndx will display the names of all index files.

The * applies to several characters. The ? wildcards individual characters. To illustrate, DIR ??SE.* will display the names of files where the filename is four characters long and ends in SE with any file extension.

**Soft Carriage Return** — A character automatically inserted at the end of a text line, which the editor regards as a carriage return. Also referred to as wrap-around.

**Sort** — To rearrange information in a logical sequence such as alphabetical or numeric order. In dBASE III PLUS, a new database file is created with records in the sorted order.

**Standard** — An attribute of the monitor display used by most dBASE III PLUS commands. On a monochrome monitor this appears as a white foreground on a black background.

**Status Bar** — A line at the bottom of the screen containing information about the current session, such as the file being used and the default disk drive.

**String** — See Character String.

**Structured Programming** — The art of writing program modules such that each module accomplishes one specific task in the program.

**Submenu** — A list of options that appear as a result of a menu selection.

**Subroutine** A program which is called by another program and dependent on it.

**Substring** A portion or subset of a string.

**Syntax** The rules governing the structure of a command.

**Template** A mask to aid, define, or limit the entry of data. The template 99/99/99 can be used as an aid in entering dates into a character variable, for example.

**TO** A part of many dBASE III PLUS commands that indicates the target of the command. TO is normally followed by either a filename or a memory variable name. Some commands use TO as a natural connector to specify a processing characteristic (for example, SET COLOR TO).

**Toggle** A switch that is always in one of two different states. For example, the cursor control menu is toggled on and off using **F1**.

**Top-Down Design** A way of designing programs starting with an initial concept and then breaking the program down into specific parts or modules.

**Trailing Blanks** The blank spaces following the last non-blank character in a character variable, to the length of the field.

**Truncate** To drop or trim digits from numbers or characters in a character string. For example, the number 4.342 can be truncated to 4.

**Turnkey System** A program which begins automatically when the user turns on the computer and which cushions the user from the actual working of the computer.

**Type-ahead Buffer** An area that stores keyboard input until dBASE III PLUS is able to respond to this input.

**Uninstall** To copy dBASE III PLUS from a secondary disk back to its original system disk. Uninstalling is done with the UNINSTAL batch program.

**User-defined Values** — Values that must be entered by the user.

**User Interface** — The software mechanism by which users access, respond to, and control an application program. The user interface includes screen forms, prompts, messages, and help.

**Variable** — A symbol with a value that may change during program execution.

**View** — A selective way to look into a set of files. A view is itself a special file that sets relations between database files and selects the fields that may be accessed.

**WHILE <condition>** — An optional clause in many commands which indicates that the command is to be repeated as long as the stated condition is True. When WHILE <condition> is used with a command that applies to database records (such as LIST), each record is evaluated in sequence and the command terminates when either the condition is False or the scope is exceeded.

**Wildcard** — A symbol that permits substitution for individual characters or groups of characters in variable names and filenames. The wildcard substitution for individual characters is the question mark (?). For groups of characters it is the asterisk (*).

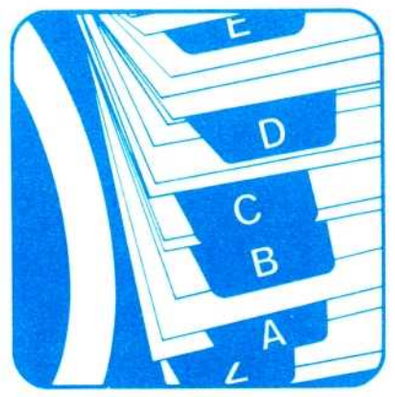
**Work Area** — dBASE III PLUS is divided into ten independent work areas, each one capable of containing one database file, its associated index files, a format file, a filter, and a linkage to another work area.

**Workstation** — A computer terminal, usually part of a Local Area Network, at which work is done. Workstations are linked to a central processor or file server.

**Working Environment** — The way an application program handles the screen, database files, and disk space during the run of the program. The working environment may be different from the default working environment of dBASE III PLUS at the dot prompt.

**Zoom**

A feature that allows constricted data entry areas to expand to 80-column width for easier reading. Zoom is toggled on by **Ctrl-PgDn** and toggled off by **Ctrl-PgUp**.




# Index


1.2 Mbyte drives, N1-8
256K Configuration, U4-10
3COM 3+ Network, N1-8, ND-1 – ND-17
& (macro substitution), U2-9, U2-10, U6-7 – U6-9
&& command, P1-22, U3-5, U5-159
? catalog query clause, U2-54, U5-1, U5-198 – U5-199, U5-201
?/?? query command, U5-15
  with memo fields, P8-17
  with printed output, P12-4, P12-5
  in programs, P5-2, P11-2
  for special printing effects, P12-5, P12-12 – P12-14
  with templates, P7-9
.AND., P15-3, U2-6
.NOT., P1-10, P15-4, U2-6
  in program flow, P2-10
  logic errors, P15-4
.OR., P1-10, U2-6
  logic errors, P15-4
  in program flow, P2-10,
@ function, in PICTURE clauses, P7-4, U5-17, U5-21
@...CLEAR, P6-7, P14-4, P14-6, U5-16
@...CLEAR TO, P6-8, P8-8, U5-23, U5-24
@...GET...SAY, U5-16 – U5-22
  activating @...GETs, U5-170
  activating a format file, U5-20, U5-232
  changing appearance, P8-1
  clearing, P6-13, U5-22, U5-47, U5-50
  confirming input, P6-13
  converting data entry to upper case, U5-21
  creating a format file, *see* CREATE/MODIFY SCREEN
  editing fields and variables, U5-16
  editing memo fields, U5-16
  horizontal scrolling, P7-6, U5-18 – U5-19
  in format files, P8-12, U5-16, U5-170
  maximum number of GETs, P6-12, U5-170
  on-screen appearance, P6-11, P8-1, P14-6
  order of screen display, P6-6
  with page ejects, P12-12, U5-16
  PICTURE function, U5-18 (table), U5-17 – U5-19
  with PICTURE templates, P7-1, U5-17 – U5-19
  with printed output, P12-3 – P12-4
  with ranges, P7-8
  with relative addressing, P8-3
  release all @...GETs, U5-50
  routed to printer or screen, U5-16, U5-218
  row and column coordinates, U5-16
  several on one line, P6-12
  specifying a RANGE, U5-17
@...TO, P8-7, U5-23 – U5-24
↵ key
  evaluating, P9-10
  in @...GET blanks, P6-13
  testing for, P9-10

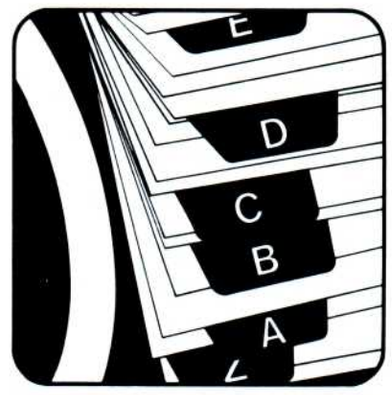

## A


Abandoning
  changes made, L3-12 (note), U2-19
  PROTECT entries, N3-31
ABS() function, P5-5, P11-1, U6-10
Absolute value, P5-5, P11-1, U6-10
ACCEPT, U5-25. *See also* INPUT, WAIT
  difference from INPUT, P6-14
Accepting character input, *see* Input, verifying
ACCESS disk, N1-7, N2-3
  on a 3Com network, ND-13
  on an IBM network, NB-13
  on a Novell network, NC-11
ACCESS() function, N5-28
Access level(s)
  establishing, N3-20, N3-26, N3-27
  field privileges, N3-5, N3-27
  file privileges, N3-5, N3-26
  for users, N3-4, N3-21
Account name, N3-19
Action line, L1-13 (figure), L1-14, U2-14
Action section, status bar, N3-13
Activating files, *see* Opening files
Adding records. *See also* APPEND, BROWSE, INSERT
  in The Assistant, L1-20 – L1-22, L3-7 – L3-9
  copying data from previous, U5-196

**Key to Index Page Numbers:**
L — Learning
N — Networking
NA-ND — Networking Appendices
P — Programming
R — Runtime
U — Using




Adding users, *see* Adduser program
ADDITIVE option, *see* RESTORE
Adduser program
  definition, N1-6, N2-3
  error messages, NA-3
  running on a 3Com network, ND-13
  running on an IBM network, NB-13
  running on a Novell network, NC-11
Administrator, *see* dBASE Administrator
Advanced Programmer's Guide, P16-17
ALIAS, P10-5, P11-11, U2-38, U5-99, U5-223 – U5-224, U5-227 – U5-228
ALL scope
  with DELETE, P11-8
  with DISPLAY, P11-3
Alphabetic key, testing for, P9-12
American Standards Committee for Information Interchange, P1-4
APPEND, U2-23, U5-26 – U5-27. *See also* SET FORMAT TO
  with active index files, U5-26
  in The Assistant, L3-3 (table), L3-7 – L3-10
  with BLANK, N4-7, N5-4, P11-6, U5-26
  format files and, P8-12, U5-26
  updating data, P11-4, P11-6
APPEND FROM, N4-7, N5-4, U5-28 – U5-31. *See also* EXPORT, IMPORT
  defaults, U5-28
  DELIMITED, U5-29
  to export/import files, P13-11, P16-14
  file types, U5-29
  records marked for deletion, U5-28
  with SET DELETED, P10-18, U5-28
Application planning, N2-4 – N2-5
Applications, AI-1, P1-2, R1-1
Applications, sample
  airline reservation network, N4-15, N4-22 – N4-25
  checkbook management system, P1-15 – P1-18, R1-4 – R1-10
Applications Generator, AI-1 – AI-2
  adding records, A4-6
  advanced features, A8-1
  changing color display, A1-3 – A1-4
  changing existing application, A7-5 – A7-6
  CREATE screen, A2-2 – A2-3
  creating an application, A3-1 – A3-3, A6-4 – A6-5, A7-1 – A7-5
  creating a database file, A2-2 – A2-7
  creating a label, A6-3 – A6-4
  creating a report, A6-1 – A6-3
  creating a screen form, A5-1 – A5-6
  deleting records, A4-7
  editing records, A4-6 – A4-7
  exiting, A1-3
  main menu, A1-2 (figure)
  reviewing records, A4-6
  running the application, A4-1
  saving file structure, A2-6 – A2-7
  selecting menu options, A1-3
  starting, A1-1
Applications program. *See also* Applications Generator
  control command file, R4-4
  with dBRUN III PLUS, R1-2
  documentation, P1-22, R1-1, R1-11
  encrypting, R2-1 – R2-6
  linking encrypted files, R3-1 – R3-6
  multi-disk, R4-4
  overlay file, R4-4
  shipping, R4-4
  starting from DOS, P16-16 (tip)
  technical support, R1-1, R1-11
Arithmetic, *see* Mathematical operations
Arranging records, *see* INDEX, Index (.ndx) files, SORT
ASC() function, P5-14, U6-11
ASCII
  displaying file, *see* TYPE
  files, importing and exporting, U2-55 – U2-58, U5-28 – U5-31. *See also* EXPORT, IMPORT
  file formats accepted, U2-57 (table)
  response file, R1-3, R1-6. *See also*Response file
  value, *see* ASC()
ASCII codes, P1-4
  for data types, UC-3 (table)
  when comparing strings, P5-4
  determining, P5-14







different meanings, P5-15
evaluating, P9-7
form feed, P12-11
for graphics, P8-7
tables, UD-1 – UD-3
to character, *see* CHR()
with @...SAY, P8-9
Assembly language routines, *see also* CALL, LOAD, RELEASE
executing, P16-10, U5-43
limitations, P16-10, U5-43
loading into memory, P16-11, U5-149
releasing from memory, P16-11, U5-173
ASSIST, U5-32 – U5-38. *See also* Assistant, The
Assistant, The, L1-6 – L1-15, L1-7 (figure).
action line, L1-13 (figure), L1-14
cancelling selections, L1-14
Create Menu, L1-15, U5-33 – U5-34
exiting to dot prompt, L1-14
(warning), L9-2
help in, L1-15
menu(s), L1-1 (figure), L1-7 (figure), L1-8 – L1-12, U5-32
menu bar, L1-8
message line, L1-13 (figure), L1-14
Modify Menu, U5-37
navigation line, L1-13 (figure), L1-14
opening menus, L1-8 –L1-9
Organize Menu, L4-12 – L4-13, L4-19 – L4-20, U5-36 – U5-37
Position Menu, to find records, L4-2 – L4-9, L4-14 – L4-17, U5-35 – U5-36
quitting dBASE III PLUS, L1-26, U5-29
Retrieve Menu, L4-9 – L4-11, L6-16, L6-24 – L6-27, U5-36
selecting menu options, L1-9 – L1-12
Set Up Menu, to open files, L3-2 – L3-3, U5-32
starting from the dot prompt, L4-14, L9-4 – L9-5
status bar, L1-13 – L1-14 (figure)
Tools Menu, L1-25, L8-2 – L8-5, U5-38
Update Menu, L3-3 (table), L3-3 – L3-17, U5-34 – U5-35
AT(), P5-8, U6-12
testing for spaces, P9-11
ATTRIB command
3Com network, ND-8
IBM network, NB-8
Novell network equivalent, NC-8
Attributes
file access, N4-15
file open, N4-2, N4-6
screen, U5-194 – U5-197, U5-232
Autoexec.bat (DOS), NB-10, ND-13, P16-15
Automatic file locking, *see* Locking
AVERAGE, U5-39. *See also* COUNT, SET TALK, SUM, TOTAL
on a network, N4-7, N5-4

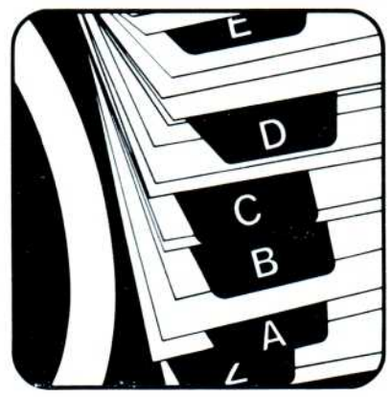

## B


Backing up
disks, NB-2, NC-3, ND-3
files, P13-10
Backup
procedures, L1-25 – L1-26, U2-59, U5-59
program files, P13-10
records from hard disk, P13-9
Backup files
database (.bak), U1-5, U5-92 – U5-93
memo (.tbk), U1-6
program, P13-10, U1-6
Batch files, P12-3, P16-12, P16-15
using RUN with, P16-12
Batch record editing, U2-25, U5-176 – U5-177, U5-279 – U5-280
Beginning-of-file, *see also* BOF()
condition, P10-13, UB-1 – UB-2 (tables)
determining, U6-13
difference from TOP, P10-13
Bell. *See also* SET BELL, SET CONFIRM
as ASCII character, P5-14
controlling in programs, P4-5
ringing in programs, P5-14
Binary (.bin) files, P16-10, U1-5, *see also* CALL, LOAD
calling, U5-43
loading, U5-149 – U5-152

releasing, U5-173 – U5-174
Blackboard, L2-5 (figure), U2-44, U5-84
*See also* CREATE/MODIFY SCREEN, Screen form(s), Screen Painter
editing keys, L2-9 (table)
field highlight, L2-7
field labels versus field names, L2-17
status bar, L2-7
Blanks
as delimiters, U5-29, U5-56. *See also* APPEND FROM, COPY
generating, *see* SPACE()
leading, P5-11. *See also* LTRIM()
trailing, L6-22, P5-10. *See also* RTRIM(), TRIM()
BOF() function, L10-6, P10-13, P10-15, U6-13
*See also* EOF()
Bottom margin, P12-8
Boxes
clearing, P8-8, U5-23
drawing, L2-30 – L2-31, P8-7, U5-19 – U5-20, U5-23 – U5-24
stretching and shrinking, L2-31
for user input, P8-8
Branching, P1-10, P2-3, U3-3. *See also* CALL, DO
conditional, in program, U5-113, U5-128
BROWSE, L9-18, N4-7, N5-4, U2-22, U2-25, U5-40 – U5-42. *See also* DISPLAY, Displaying
in The Assistant, L3-3 (table), L3-4 – L3-7
adding records in, L3-6
in application programs, P6-2, P11-4, P11-16
command line options, U5-40 – U5-41
editing and appending records in, U5-40
menu bar, L3-5 (figure), L3-6 (table), U5-41 – U5-42
on a network, N4-7, N5-4
BUCKET, *see* Configuration commands
Buffer
history, L9-6, U5-107, U5-145, U5-219, U5-237 – U5-238
print, P12-11, U5-107, U5-145
type-ahead, U5-48, U5-263

## C

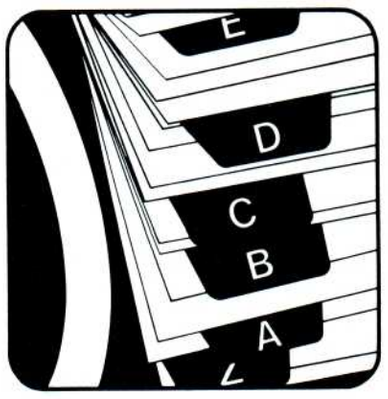
-c option, R2-4, R3-4
Calculation commands, *see* AVERAGE, COUNT, Mathematical functions, SUM, TOTAL, ?/??
CALL, P16-10, U5-43. *See also* LOAD
maximum number of binary files, U5-43
with parameters, U5-43
Calling program, P2-3
CANCEL, P15-8, P15-12, U5-44
Cancelling
changes to record, L3-12 (note)
file privilege scheme, N3-30
menu selections, L1-14
Case conversions, L4-17, P5-8. *See also* LOWER(), UPPER()
Catalog (.cat) file(s), L7-11, U1-5, U2-49 – U2-54. *See also* SET CATALOG
activating, L7-13, U2-52, U5-29, U5-197 – U5-198
adding entries, L9-11 – L9-13, U2-52, U5-201
changing, L7-14
changing entry names, U2-53
closing, U2-53, U5-198
creating in The Assistant, L7-13 (note)
creating at the dot prompt, L9-9 – L9-10, U2-52, U5-197 – U5-198
deleting entries, U2-53, U5-198
fields, U2-51, U5-199
file title prompt, U2-51, U5-197 – U5-198
master (catalog.cat), U2-51, U5-197
opening, U2-52, U5-197
query (?) clause, U2-54, U5-198
renaming entries, U2-53
selecting in The Assistant, L7-13, L9-10 – L9-13
selecting a file from, L9-13, U5-198
structure, U2-51, U5-199
CDOW() function, P5-22, U6-14. *See also* DOW()
Century, *see* SET CENTURY
changing, P5-22
prefix, U5-203






CHANGE, N4-13, N5-6, P6-2, P8-12, P11-4, U5-41 – U5-42, U5-45. *See also* EDIT
  with memo fields, P8-16
  on a network, N4-13, N5-5
Changes from dBASE II. *See dBASE BRIDGE*
Changing records, *see* CHANGE, EDIT, Editing records
Character
  field, U1-9, U5-89
  manipulation functions, U6-5 (table)
  memory variable, P3-4, U2-10
Character string
  centering, P8-5 – P8-6
  converting to ASCII, P5-14. *See also* ASC(), CHR()
  converting date to, *see* DTOC()
  converting to date, *see* CTOD()
  converting to lower case, *see* LOWER()
  converting a number to, *see* STR()
  converting to numeric, P15-19, U6-89 – U6-90. *See also* VAL()
  converting to upper case, *see* UPPER()
  determining case, *see* ISLOWER(), ISUPPER()
  extracting characters from, P5-8. *See also* LEFT(), RIGHT(), SUBSTR()
  formatting, *see* TRANSFORM()
  functions, P5-5 – P5-19
  inserting one, within another, P8-6 – P8-7. *See also* STUFF()
  length of, P5-6, U6-49. *See also* LEN()
  operators, L5-8 (table), U2-5 – U2-6
  position of substring, P5-8. *See also*AT()
  removing leading blanks, *see* LTRIM()
  removing trailing blanks, U6-74, U6-86. *See also* RTRIM(), TRIM()
  repeating, P8-10, U6-70. *See also* REPLICATE()
  replacing part of, P8-6. *See also* REPLACE, STUFF()
  returning first character's ASCII value, *see* ASC()
  right justifying, P8-6
  show starting position within another string, *see* AT()
  substrings, P5-7
Checkbook management system
  diagram, P1-17
  list of files, PI-4, R1-7, R1-10
  RunTime + files, R1-10
  starting the program, P1-14
CHKDSK (DOS), P16-12
CHR() function, P5-14, P8-9, U6-15 – U6-16. *See also* ASC()
Chronological order, *see* INDEX, SORT
Classes of commands, N5-2, U5-3 – U5-14
  Addition of Data, U5-4
  Creation of Files, U5-4
  Data Display, U5-6
  Debugging, U5-13
  Deletion of Data, U5-5
  Editing of Data, U5-5
  Environmental, U5-10
  Event Processing, U5-14
  External Program Interfacing, U5-13
  Manipulating Database Files, U5-7
  Manipulating Other Types of Files, U5-7
  Parameter Control, U5-10 – U5-12
  Positioning the Record Pointer, U5-6
  Programming, U5-9
  User Assistance, U5-3
  Using Memory Variables, U5-8
Classes of functions, U6-4 – U6-6 (table)
CLEAR, L10-3, P6-7, P6-13, U5-47. *See also* @, CLEAR GETS
  erasing the screen, U5-47
  releasing pending GETs, U5-50
CLEAR ALL, U5-48. *See also* CLEAR MEMORY, CLOSE, RELEASE ALL
  to close database files, P4-2
  leaving a program with, P2-2
  to release PUBLIC variables, P3-10
  to unlock file records, N4-12
CLEAR FIELDS, P11-16, U5-49. *See also* SET FIELDS, SET VIEW
CLEAR GETS, P6-13, U5-50
CLEAR MEMORY, P3-10, P13-3, U5-51
CLEAR TYPEAHEAD, P8-17, U5-52. *See also* SET TYPEAHEAD TO

Clearing
a box, U5-23 – U5-24
memory, P13-3
the screen, P6-6.*See also* @...CLEAR, @...CLEAR TO, CLEAR
the type-ahead buffer, *see* CLEAR TYPEAHEAD
CLOSE, N4-12, U5-53
CLOSE DATABASES, L9-3, P2-2, P10-4, P11-7, P13-2, U5-53
CLOSE FORMAT, P8-14
CLOSE INDEX, P10-7
CLOSE PROCEDURE, P16-4, U3-11
Closing files, L1-25, L9-30, P13-2, U2-33, U2-35 (table). *See also* CLEAR ALL, CLOSE, USE
Cluster networks, N2-4
CMONTH() function, P5-22, U6-17. *See also* MONTH()
COL() function, P8-3, U6-18. *See also* PCOL(), ROW()
Collision, N4-2, N4-10
Color monitors, P4-5, U2-20, U5-204 – U5-208
Color, testing for. *See* ISCOLOR()
Column(s), *see also* CREATE/MODIFY REPORT
coordinate, P6-4
heading in report, L6-10 – L6-13
inserting into report, L6-14 – L6-15, U5-78
layout in reports, L6-10 – L6-15, U5-78 – U5-79
totals in report, L6-13, U5-79
Command(s), U2-1, *see also* Classes of commands
abbreviating, L9-4, U2-7
branching, P2-33, P2-34, U3-3
commonly used, L9-8 – L9-9 (table)
conditional execution of, U5-114 – U5-117, U5-128
configuration, U4-3 – U4-5
correcting, L9-3
editing data, L9-22, U5-5
entering from dot prompt, L9-3 – L9-6, U2-1 – U2-2
expression list, in U2-3
format file, L9-16 – L9-17
full-screen, U2-20
history, U2-2, U5-107, U5-145, U5-219, U5-237 – U5-238
to invoke dBASE ADMINISTRATOR, N2-6
length of, L9-21
memory variable, L9-26 – L9-29, U5-8
network programming, N5-1 – N5-3
not supported in RunTime +, R4-1
programming, U3-2 – U3-4, U5-9
re-entering, U2-2
requiring exclusive use, N5-4
requiring a lock, N5-4
reserved names, U2-8
rules for writing, U2-7 – U2-8
scope, U2-3
scrolling, L9-21
structure, U2-3
syntax, U2-3
verb, U2-3
Command files, L10-1 – L10-4, U1-5
*See also* Program (.prg) files, MODIFY COMMAND
closing, U5-44
creating, L10-1 – L10-3, U5-155 – U5-157
executing, U5-106
modifying, L10-3, U5-155 – U5-157
stopping execution of, U5-44
Command line
Insert and Overwrite modes, L9-3
length, L9-21, R3-6, U2-7
dBCODE formats, R2-1 – R2-6
dBLINKER formats, R3-1 – R3-6
options with dBCODE, R2-4
options with dBLINKER, R3-3
specifications, U1-1 – U1-2
Command processor (DOS), P16-13
Command.com (DOS), N2-10, P16-13, N5-22, U4-6, U5-185
Comment lines, L10-3, P1-23, U5-159
Comparisons
between different types, P5-4
exact, P10-18
COMSPEC (DOS), P16-13, U5-185



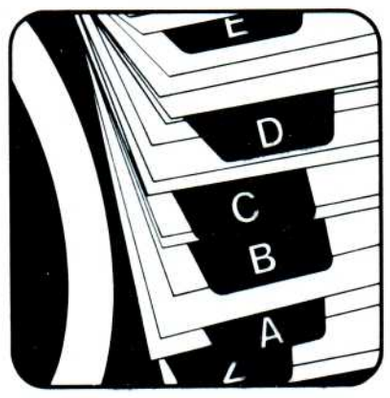



Concatenation, P5-3, P5-13. *See also* String operators
Condition(s), L4-7, U2-3
  for determining input, P9-4
  in program flow, P2-10 – P2-12
  scope, L4-18 – L4-19
  search, L4-4 – L4-7, L4-10 – L4-11, L5-2 – L5-9
  testing for, P14-3
Confi256.db, U4-10
Confi256.sys, U4-10
Config.db, U4-2
  at a workstation, N1-13
  commands and values, U4-9 – U4-10 (table)
  creating, U4-7 – U4-8
  dBASE configuration commands, U4-2, U4-4 – U4-6
  with default drive, P16-16
  with default settings, P4-3
  function keys in, U4-7
  modifying, U4-7 – U4-8
  with RUN command, P16-12
  SET commands in, U4-6
  storing, U4-3, U4-8
  on System Disk #1, U4-2
  256K configuration, U4-10
Config.sys, U4-1
  at a 3Com workstation, ND-3
  creating, U4-2
  DOS configuration commands, U4-1
  FILES parameter, U2-33, U4-1
  at an IBM file server, NB-3
  at an IBM workstation, NB-4
  modifying, U4-2
  at a Novell workstation, NC-4
  storing, U4-2
  on System Disk #1, U4-1
Configuration commands
  BREAK (DOS), U4-1
  BUCKET, U4-4
  BUFFERS (DOS), U4-1, U5-234
  DEVICE (DOS), U4-1
  FILES (DOS), U4-1
  for function keys, U4-7
  GETS, U4-5
  MVARSIZE, U4-5
  PROMPT, U4-6
  SET commands, U4-6. *See also* individual SET entries
  SHELL (DOS), U4-1
  TEDIT, U4-6
  WP, U4-6
Configuration file, *see* Config.db, Config.sys
CONTINUE, P10-9, U5-54
  in The Assistant, L4-6 – L4-7
Continuous loop, P4-11
Control characters, UD-3 (table)
Conventions, *see* Symbols and conventions
Conversion
  functions, U6-5 (table)
  guidelines, P5-3
Converting
  case, *see* LOWER(), UPPER()
  character to date, *see* CTOD()
  data types, P5-2
  date to character, *see* DTOC()
  files from dBASE II to dBASE III PLUS format, *see dBASE Bridge*
  input, with templates, P7-2, ,U5-18
  number to character, *see* CHR(), STR()
  numeric expression to integer, *see* INT(), ROUND()
Coordinates, P6-4. *See also* COL(), PCOL(), PROW(), ROW()
  limits for printed output, P12-7
  numbering, P6-4
  printer, P12-4
  with relative addressing, P8-3
COPY, U2-58, U5-55 – U5-58
  in The Assistant, L1-25 – L1-26, L4-21 – L4-22
  encrypted (.crp) files, N5-15
  on a network, N4-7, N5-4
  with other programs, P16-14
  scratch file, P13-10
COPY FILE, U2-59, U5-59
COPY STRUCTURE, N4-7, N5-4, U5-60
COPY STRUCTURE EXTENDED, P13-11, U5-61 – U5-62
  with encrypted files, N5-16

**Key to Index Page Numbers:**
L — Learning
N — Networking
NA-ND — Networking Appendices
P — Programming
R — Runtime
U — Using



Copying
encrypted files, N5-15
fields, L4-21 – L4-22, U5-58
file structure, U5-60, U5-61 – U5-62
files, L1-25 – L1-26, P13-10, U2-58 – U2-59, U5-55 – U5-58, U5-59
files, foreign, *see* APPEND FROM, COPY, EXPORT, IMPORT
memo fields, U5-55
records, P11-10
records marked for deletion, U5-55
Copyright notice
in applications programs, R1-8, R1-9, R2-4, R3-4
COUNT, N4-7, N5-4, U5-63
Counter
for page ejects, P12-16
setting up, P11-2
defining update interval of, U5-246
Counting records, *see* RECCOUNT()
CPUs supported on networks, N1-7
CREATE FROM, P13-11, P13-14, U5-65. *See also* COPY STRUCTURE EXTENDED
CREATE VIEW FROM ENVIRONMENT, P11 – 15, U5-101
CREATE/MODIFY LABEL, U2-43, U5-66 – U5-69. *See also* Label(s), LABEL FORM, Label form (.lbl) files.
in The Assistant, L6-17 – L6-24, U5-30
Contents Menu, L6-21 – L6-24, U5-68 – U5-69
menu bar, L6-18 (figure)
Options Menu, L6-18 – L6-19, L6-20 (table), U5-66 – U5-67
predefined sizes, U5-67 (table)
zoom option, U5-68
CREATE/MODIFY QUERY, U5-70 – U5-72. *See also* Query (.qry) files, SET FILTER
in The Assistant, L5-1 – L5-17
menu bar, L5-3 (figure)
Nest Menu, L5-10, U5-71 – U5-72
Set Filter Menu, L5-5 (figure), U5-71
CREATE/MODIFY REPORT, L6-4 (figure), L9-24, U2-42, U5-73 – U5-81. *See also* REPORT FORM, Report form (.frm) files
in The Assistant, L6-4 – L6-16
Columns Menu, L6-10 – L6-15
Groups Menu, L6-8 – L6-9, L6-10 (table), U5-77 – U5-78
Locate Menu, L6-14, U5-79
Options Menu, L6-5 – L6-6, L6-7 (table), U5-75 – U5-76
page layout codes, U5-80 (table)
report option ranges, U5-81 (table)
CREATE/MODIFY SCREEN, L9-16 – L9-17, P8-12, U2-44 – U2-45, U5-82 – U5-87. *See also* Blackboard, Screen forms, Screen Painter, SET FORMAT TO
blackboard, U5-84
Screen Painter, U5-82
CREATE < newfile > /MODIFY STRUCTURE, L9-10 – L9-12, N5-4, U5-88 – U5-93. *See also* Creating, Modifying
with encrypted (.crp) files, N5-16
CREATE/MODIFY VIEW, P11-15, U2-41, U5-84 – U5-100. *See also* SET RELATION, SET VIEW TO, View (.vue) files
in The Assistant, L7-3 – L7-10, U5-30
definition, U1-7
Options Menu, L7-8
Relate Menu, L7-5 – L7-6, U5-95
Set Fields Menu, L7-7 – L7-8, U5-98
Set Up Menu, U5-95 – U5-98
Creating
an application program, *see* Applications Generator
catalogs, *see* Catalog (.cat) files, SET CATALOG TO
data entry forms, *see* Screen forms
database files,*see* CREATE/MODIFY STRUCTURE, Database files
labels, *see* CREATE/MODIFY LABEL, Label (.lbl) files
new files from existing, U2-57 – U2-59
program files, *see* MODIFY COMMAND, Program (.prg) files
query (filter) conditions, *see* SET FILTER TO



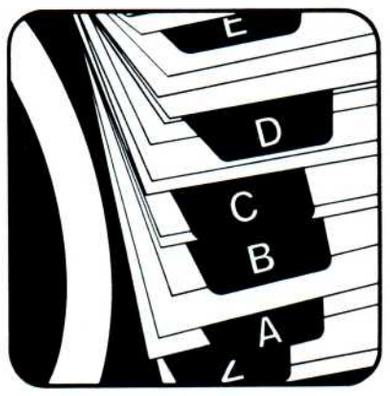



query (filter) files, *see* CREATE/MODIFY QUERY, Query (.qry) files, SET FILTER TO
reports, *see* CREATE/MODIFY REPORT, Report (.frm) files
screen forms, *see* CREATE/MODIFY SCREEN, Screen forms
view files, *see* CREATE/MODIFY VIEW, CREATE VIEW FROM ENVIRONMENT, View (.vue) files
CTOD( ) function, P5-23, U6-19. *See also* DTOC( )
for initializing variables, P5-24
Cursor control keys, U2-2 (table)
controlling, P16-10
in full-screen operations, UA-1 – UA-3
in menus, U2-15
in MODIFY COMMAND, U5-155 – U5-156
in Screen Painter, L2-9 (table)
Cursor position functions, *see* COL( ), ROW( )
Customizing
dBASE III PLUS, N1-11 – N1-12, U4-1 – U4-9
routines, P16-5
screen forms, *see* @, CREATE/MODIFY SCREEN, Screen form

## D

Data
access, simultaneous, N4-2
backing up, U2-59
catalogs, *see* Catalog (.cat) files, SET CATALOG
integrity, N1-5
merging from two files, U2-57
protection on a network, N4-1 – N4-2. *See also* dBASE security, PROTECT
relating, U2-38 – U2-40, U5-94 – U5-101, U5-255 – U5-256. *See also* CREATE/MODIFY VIEW, SET RELATION, View (.vue) files
summarizing, L6-24 – L6-26
type, evaluating, *see* TYPE( )
Data commands, U5-2 – U5-4
Data encryption, N3-7. *See also* dBASE security, PROTECT, SET ENCRYPTION
for applications programs, *see* dBCODE, dBLINKER, RunTime +
Data entry, *see* APPEND, CHANGE, EDIT
Data entry forms, *see* Screen forms
Data integrity, P1-9, P3-7, P6-2, P10-3
Data type
converting, *see* Converting
in database fields, L1-5 (table), U1-8 – U1-9
in memory variables, L9-27, U2-10
mismatches, P15-2
Database file(s)
adding fields, L1-5 – L1-20, L2-22 – L2-25
adding records, L1-20 – L1-22, U2-22, U5-26 – U5-27, U5-137
arranging records, *see* INDEX, Index (.ndx) files, SORT
changing field width, L2-19 – L2-21
closing, L1-25, L9-30, P10-3 – P10-4, P13-2, U2-35, U5-49
combining, P11-7
copying, L1-25 – L1-26, P13-10, U2-58 – U2-59. *See also* COPY
copying fields, L4-21 – L4-22
creating in The Assistant, L1-15 – L1-20
creating at the dot prompt, L9-10 – L9-11, U2-22
creating new from old, U5-61 – U5-62, U5-134 – U5-136
currently selected, P13-5. *See also* DBF( )
deleting, P13-10, U5-97, U5-121
deleting all records from, *see* ZAP
deleting records from, L3-13 – L3-17. *See also* DELETE, PACK, RECALL
designing, L1-4 – L1-5, P10-1
determining size of, P13-7
displaying records, L1-22 – L1-24, L3-4 – L3-7, L9-18 – L9-21, U5-105
editing records, L3-11 – L3-12, U2-24 – U2-26, U5-118, U5-169 – U5-170
exporting, L8-4 – L8-6, U2-57 – U2-58
finding records in, L4-1 – L4-11
importing, L8-2 – L8-4

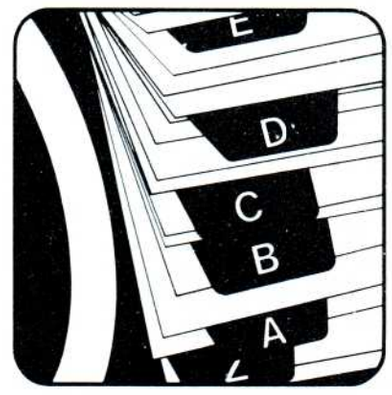
indexing, L4-12 – L4-13, L9-22, P10-6, U2-28 – U2-30, U5-131 – U5-134. *See also* INDEX, Indexing, SORT
integrity of, P1-9, P3-7, P6-2, P10-3
determining last update, P13-6. *See also* LUPDATE()
maintaining, P13-10
managing, P10-8
maximum open at one time, U2-33
modifying structure, L9-12, P13-11, U2-22. *See also* COPY STRUCTURE EXTENDED
opening, L3-2 – L3-3, L9-13, P10-3, U2-34 (table), U5-281
opening more than one, P11-11, U2-36
organizing, U2-26 – U2-30. *See also* INDEX, SORT
printing records, L6-1 – L6-30. *See also* CREATE/MODIFY REPORT, DISPLAY, LIST
related, P11-12
relating, L7-3 – L7-8, P10-2, P10-4, P11-12, U2-38 – U2-40, U5-94 – U5-101, U5-255 – U5-256, U5-266 – U5-267. *See also* SET RELATION, View (.vue) files
relational, P10-2
renaming, P13-10
requesting filename, P10-8
restricted access to, P10-3
restructuring for programs, P10-3
saving data, L1-25
searching for data, L4-1 – L4-11, U2-30 – U2-32. *See also* FIND, SEEK
selecting, L3-2 – L3-3, L9-13 – L9-14
size, P13-6
sorting, L4-19 – L4-20, L9-23, L2-29 – U2-30, U5-269 – U5-270
specifications, U1-1 – U1-2
status, L9-14, L9-15 (figure), U5-104, U5-141. *See also* DISPLAY STATUS, LIST STATUS
structure, L1-16 – L1-19, L9-15 (figure), U1-8, U5-88 – U5-90. *See also* DISPLAY STRUCTURE, LIST STRUCTURE
structure of file header, UC-1 – UC-2 (table)
updating, L3-1 – L3-19, P11-4, P11-7
Database manipulation commands, U5-7
Database memo (.dbt) files,*see* Memo (.dbt) files
Database structure, L1-16, U1-8. *See also* CREATE < newfile > /MODIFY STRUCTURE
copying, U5-57 – U5-58
creating, U5-60 – U5-65, U5-66
header, UC-1 – UC-4
memo, UC-3 – UC-3
records, UC-2
Date field, U1-8. *See also* Dates
Date functions, U6-4 (table)
Date memory variable, U2-10
DATE() function, P3-5, P5-20 – P5-25, U6-20. *See also* TIME()
initializing variables, P5-25
Dates
century prefix, U5-193
in comparisons, P5-24 – P5-25
in conversions, P5-23 – P5-24
converting from strings, *see* CTOD(), DTOC()
converting to strings, P5-22 – P5-23, U6-26. *See also* DTOC()
formats, P5-21
output format, U5-200
Day, from a date, P5-21
Day of month, determining, *see* DAY()
Day of week determining, *see* CDOW(), DOW()
DAY() function, P5-20, U6-21. *See also* CDOW(), DOW()
DBA command
on a 3Com network, ND-10
on an IBM network, NB-10
on a Novell network, NC-11
dBASE ADMINISTRATOR
with an application program, NB-11, NC-10, ND-11
definition, N1-5, N2-2
error messages, NA-5
installing on a network, NB-1, NC-1, ND-1
in a multiple file server network, N2-3 – N2-4
requirements, N1-8






in single-user environment, N2-2
uninstalling from a network, NB-15, NC-13, ND-15
dBASE ADMINISTRATOR directory
creating for 3Com network, ND-4
creating for IBM network, NB-5
creating for Novell network, NC-4
dBASE II, converting to dBASE III PLUS, *see dBASE Bridge*
dBASE III PLUS LAN Pack, NB-13, NC-11, ND-13
dBASE III PLUS word processor, P1-5, U3-8, U4-6
dBASE network commands
automatic file locking, N5-4
classes of, N5-2 (table)
guidelines, N3-33
programming, N4-21, N5-3 (table)
requiring exclusive use, N5-4
requiring lock functions, N5-4
dBASE programming language, P1-2
dBASE security
access levels, N3-4 – N3-5
creating system, N3-9
data encryption, N3-1, N3-6
Dbsystem.db file, N3-8, N3-32 – N3-33
field access, N3-1, N3-5. *See also* Field access privileges
file access, N3-1, N3-5. *See also* File access privileges
guidelines, N3-36
keeping a record of, N3-32
log-in, N3-1 – N3-4
from network applications programs, N4-19
network, keeping record of, N3-22 – N3-33
programming, N4-20
request form, N3-35 – N3-36
types, N3-1 – N3-2, N3-2 (table)
user access levels, N3-4. *See also* PROTECT
dBCODE, R1-2, R2-1 – R2-6
command line formats, R2-1, R2-2, R2-3
command line options, R1-8, R2-3
copyright header file, R2-4
encrypting files, R1-8, R2-1 – R2-6
file size restrictions, R4-3
help screen, R2-3
information file, R2-5
response file, R2-5
sample session, R1-4 – R1-10
DBF() function, P13-5 – P13-6, U6-22. *See also* NDX()
dBLINKER, R1-2, R3-1 – R3-6
command line formats, R3-2, R3-3
command line options, R3-3 – R3-6
copyright header file, R3-4
help screen, R3-3
information file, R3-5
linking files, R1-9, R3-1 – R3-6
response file, R3-6
root files, R4-4
sample session, R1-9
DBNETCTL.300 directory
definition, N1-7, NB-7, NC-8, ND-8
uninstalling from a 3Com file server, ND-17
uninstalling from an IBM file server, NB-16
uninstalling from a Novell file server, NC-14
dBRUN III PLUS, R1-2, R4-1, R4-2
commands not included in, R4-1 – R4-2
purchasing copies, R1-2, 4-1
Dbsystem.db file, N3-8, N3-32 – N3-33
Deadlock, N4-9
Debugging programs, P15-1
with disk file, P15-12
interactive, P15-8
with modular programs, P15-5
with the printer, P15-9
program files, U3-9 – U3-10
stepping through the program, P15-9, U5-252. *See also* SET STEP
suspend program execution for, U5-274, U5-182
Decimal places
controlling display of, P5-18
in conversions, P5-16
fixing number of, P5-18
in memory variables, P3-5
setting, U5-213, U5-231

DELETE, P13-13, U2-25, U5-102. *See also* ERASE, PACK, RECALL
in The Assistant, L3-13 – L3-17
on a network, N4-7, N5-4
Delete file privilege, N3-5, N3-26
Deleted records
bypassing, P10-18
reinstating, L3-14, U2-25, U5-103, U5-164
DELETE FILE, U5-97
DELETED( ) function, P10-18, U6-23. *See also* SET DELETED
Deleting
all records, U2-25, U5-284
data commands, U5-5
fields, L2-25
files, P13-10, U5-121
leading blanks from string, *see* LTRIM( )
marking records for, U2-25, U5-102. *See also* DELETED( ), PACK, RECALL, SET DELETED
memory variables, U5-173 – U5-174
multiple records, L3-14 – L3-17, P11-8, U5-102
single record, L3-13 – L3-14, P11-8
trailing blanks from string, *see* RTRIM( ), TRIM( )
user profile, N3-23
Delimiters
ASCII, U5-25
changing, P8-1, U4-10
in Config.db, U4-7, U4-10
for field widths, U5-216 – U5-217
Dialogue box, U2-12
DIF files
exporting, U2-55 – U2-57, U5-55 – U5-57
importing, U2-55 – U2-57, U5-28 – U5-31
DIR, P16-12, U5-103 – U5-104
Directory
destination, R2-5
development, R4-6
listing, P16-12, U5-34, U5-103 – U5-104
path, P4-8
source, R1-5, R4-4,
Disk access time, P10-3, P16-3
Disk drive, default, P4-6
DISKSPACE( ) function, P13-8, P13-9, U6-24. *See also* RECSIZE( )
DISPLAY, P11-2, U5-105 – U5-106
in The Assistant, L3-3 (table)
with memo fields, P8-17
DISPLAY HISTORY, P15-7, P15-10, U2-2, U5-107. *See also* LIST HISTORY, SET HISTORY
DISPLAY MEMORY, P15-11, U5-108
DISPLAY STATUS, L9-14 (figure), P15-11, U5-109
on a network, N2-6, N5-7
DISPLAY STRUCTURE, L9-15 (figure), U5-110
DISPLAY USERS, N2-8, N5-9
before uninstalling 3Com network, ND-16
before uninstalling IBM network, NB-15
before uninstalling Novell network, NC-14
Display user count option (for Adduser program)
on a 3Com network, ND-14
on an IBM network, NB-13
on a Novell network, NC-12
Displaying
data commands, U5-4
data with templates, P7-9
fields in view files, P11-16
history, L9-6
records in The Assistant, L3-4 – L3-7, L3-11 – L3-12
records from the dot prompt, L9-18 – L9-21. *See also* BROWSE, DISPLAY, EDIT, LIST
records, in screen form, L2-1 – L2-30
structure, L9-15 (figure). *See also* DISPLAY STRUCTURE, LIST STRUCTURE
user-created message, U5-245
DO, P1-6, U5-106, U5-111 – U5-112
for branching, P1-10, P2-3
difference from DO WHILE...ENDDO, P2-6
with ON command, U3-4, R4-2
with parameters, P16-7
with PROCEDURE files, P16-4






DO CASE...ENDCASE, P1-14, P2-10, U3-3, U5-113
  difference from IF...ENDIF, P2-12
  in debugging, P15-9
DO WHILE...ENDDO, P2-4, P1-11, P14-2, P14-5, U3-2, U5-110 – U5-114 – U5-117
  for continuous loop, P4-11
  difference from DO, P2-6
  difference from IF...ENDIF, P2-7
  EXIT, P2-17, P4-12, P14-3
  LOOP, 2-16
  with macro substitution, P3-16
DO...WITH, P16-8
Do.com, P16-16
DOS
  command execution, U5-178
  configuration commands, U4-1. *See also* Config.db, Config.sys
  device, set for printing, U5-218, U5-251 – U5-252
  exiting to, L1-26, L9-30, U5-162
  path, NB-10, ND-10
  printer port names, P12-2
  text files (Framework II), P1-4
  version and network software, N1-7
Dot prompt. *See also* Command line
  displaying from The Assistant, L9-2
  entering commands, L9-3 – L9-5, U2-1 – U2-2
  returning to, L9-2, P13-4, U4-5
  starting dBASE at, L9-2 (note)
Dot prompt line, *see* Command line
DOW() function, P5-21, U6-25. *See also* CDOW()
Drawing
  boxes, *see* Boxes, drawing
  lines, *see* Lines, drawing
Drive, set default, U5-203
DTOC() function, P5-22, U6-26. *See also* CTOD()
Duplicate records
  ignoring, P10-19
Duplicating
  any file type, U2-59, U5-59
  ASCII files, U2-59, U5-55 – U5-58
  database files, L1-25 – L1-26, U2-58, U2-59, U5-55 – U5-58
  fields, L4-21 – L4-22
  file structure, U5-56, U5-61 – U5-62

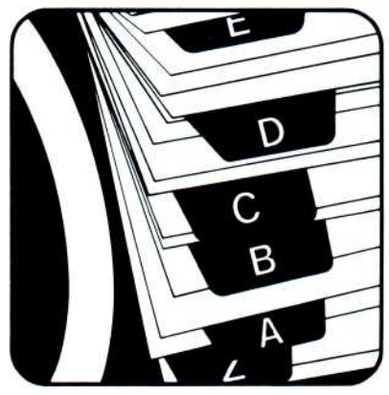
## E

EDIT, L9-22, P6-2, P8-12, P11-4, U5-118 – U5-119. *See also* CHANGE
  in The Assistant, L3-11 – L3-12
  on a network, N4-13, N5-6
Editing
  commands, U5-5
  keys, L3-4, U2-2
  labels, L6-25, U5-66 – U5-69
  in MODIFY COMMAND, U5-155 – U5-157
  records, L3-11 – L3-12, L9-22, U2-24 – U2-25, U5-114. *See also* BROWSE, CHANGE, EDIT
  report form, L6-16,U5-73 – U5-81
EJECT, P12-11, P12-12, U5-120
ELSE, *see* IF...ENDIF
Encrypted (.crp) files, N3-1, N3-6. *See also* dBASE security, PROTECT, SET
ENCRYPTION
  creating, N3-6
  and Dbsystem.db, N3-8
Encrypted (.prg) files. *See also* dBCODE, dBLINKER
  creating, R2-1 – R2-6
  linking, R1-9, R3-1 – R3-6
  named in a response file, R3-6
ENDIF, U5-123
End-of-file
  condition, P10-14, UB-1 – UB-2 (table)
  difference from BOTTOM,
  testing for, *see* EOF()
Entering data, *see* Data entry
Enumerated data, N3-16, U2-17
Environment
  dBASE III PLUS, U4-1 – U4-10
  network, N1-2 – N1-3
  operating system, P16-13
Environmental commands, U5-8

EOF() function, P10-13, U6-27. *See also* BOF()
ERASE, P13-10, U5-121
Erasing, *see* Clearing, Deleting
ERROR() function, U6-28 – U6-29, U7-1. *See also* MESSAGE(), ON ERROR
  in network programming, N4-16, N5-30
  during program run, P15-13, P16-9
Error messages. *See also* ERROR(), MESSAGE(), ON ERROR
  Adduser program, NA-3
  dBASE ADMINISTRATOR, NA-5
  IDLAN program, NA-1
  installation, NA-2
  list, U7-1 – U7-21
  when programming, P16-9
  RunTime +, RA-1
  uninstallation, NA-2
Error trapping, N4-16 – N4-20, P16-9. *See also* ON ERROR
  at the dot prompt, N4-16
  in a program, N4-16, N4-17, P9-1 – P9-4
**Esc** key
  to cancel selections, L1-14
  for debugging, P15-11
  in dBRUN III PLUS, R4-2
  with ON command, P9-12 R4-2
Escape codes, U5-15, U6-15 – U6-16
Evaluating expressions, *see* ?, ?? commands
Exact comparisons, P10-18
Executing
  binary files, *see* CALL, LOAD, RELEASE
  command file or procedure, *see* DO, PROCEDURE, SET PROCEDURE
Exiting
  dBASE III PLUS, L1-26, L9-30, U5-169
  The Assistant, L1-14 (warning), L9-2
EXP() function, P5-5, U6-30
Explicit file locking, N4-8
Exponents, *see* EXP()
EXPORT, U5-122 – U5-123.
  in The Assistant, L8-4 – L8-6
  pfs:File, U5-122 – U5-123
  screen form options, L8-6
Exporting files, *see* EXPORT
Expression(s), U2-4
  evaluating, L9-25 – L9-26
  evaluating data type, *see* TYPE()
  with index files, P10-6, U5-2, U5-131
  limitations, U5-2
  list, L9-20
  nesting, L5-10 – L5-11
  with SEEK, P10-10, U5-188
  types, U2-4
Extend file privilege, N3-5, N3-26
Extensions, file, *see* File extensions

## F

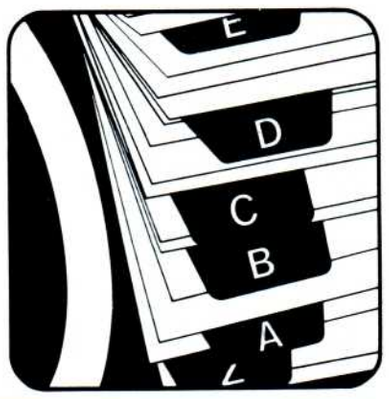
-f option, R3-4
Field(s)
  access levels, N3-27
  access privileges, *see* Field access privileges
  adding new, L1-15 – L1-19, L2-22 – L2-24
  changing contents, U2-22. *See also* BROWSE, CHANGE, EDIT, REPLACE, UPDATE
  changing width, L2-19 – L2-21
  copying selected, L4-21 – L4-22, U5-28 – U5-31, U5-55 – 58
  definition, L1-3, L1-4
  deleting, L2-25
  displaying column headings above, U5-226
  displaying contents of, *see* Displaying records
  list, N3-30 – N3-31. *See also* SET FIELDS
  moving, L2-12 – L2-17
  names, *see* Field names
  release all, U5-49
  replace, U5-176 – U5-177
  selecting, U5-212 – U5-218. *See also* SET FIELDS
  set automatic advance, U5-209
  size limitations, U1-1
  types, *see* Field types
  in view files, L7-5 – L7-8, U5-92
  width, *see* Field width
FIELD() function, P13-6, P13-8, U6-31 – U6-32






Field access privileges
  access levels, N3-27
  establishing, N3-27 – N3-29
  types, N3-27
Field names
  determining, *see* DISPLAY STRUCTURE, FIELD()
  precedence over memory variables, U2-9
  requirements, L1-4, U1-8, U5-2
Field types, L1-4 – L1-5 (table), U1-8 – U1-9
Field width
  changing, L2-19 – L2-21
  definition, L1-4, U1-8
  entering, L1-17 – L1-18
  indicated in full-screen mode, U5-205
File(s). *See also* individual file types
  access attribute, N4-15
  access level, N3-26
  Autoexec.bat, NB-10, ND-13, P16-15
  closing, L1-25, L9-30, U2-35, U5-48, U5-53
  concurrent use of, N4-2
  Config.db, N3-3, U4-2
  Config.sys, N1-11, N1-12, N1-27, U4-1
  converting from other formats, L8-2 – L8-4, U5-28 – U5-31, U5-55, U5-129
  copying any, L1-25, U2-59, U5-59
  copying database, U2-59, U5-55 – U5-58
  copying structure, U5-60, U5-61 – U5-62
  creating, U5-83, U5-88 – U5-92
  creating in The Assistant, L1-15 – L1-20, U5-33
  creation commands, U5-2
  database, defined, L1-2 – L1-3
  Dbsystem.db, N3-8, N3-32 – N3-33
  deadlock, N4-9
  deleting, U5-34, U5-97, U5-116
  determining existence, *see* DIR, FILE()
  determining size, P13-6
  exporting and importing, *see* COPY, EXPORT, IMPORT
  extensions, *see* File extensions
  group, N3-7, N3-26
  local, N2-5
  locking, *see* Locking
  Login.db, N1-11
  maintenance, P13-10
  management, P10-8
  maximum number open, U2-33
  merging data, U2-57
  naming conventions, P1-15, U5-2
  non-network, N2-5
  open attributes, *see* File open attributes
  opening database, L3-2 – L3-3, L9-13 – L9-14, U2-33 – U2-34
  opening database and index, L4-14, L9-22, U5-281
  opening, on a network, N4-2 – N4-6
  operations, specifications, U1-1
  protecting, U2-60, U5-51 – U5-54, U5-55
  relating, *see* CREATE/MODIFY VIEW, SET RELATION, View (.vue) files
  renaming, U5-34, U5-59, U5-175
  security, N3-1
  selecting, *see* Selecting
  SET PATH TO, U5-249 – U5-250
  sharing, N1-4
  size, R2-2
  types, L9-16 (table), U1-4 (table), U1-5 – U1-8. *See also* individual types under . (period)
  using simultaneously, U2-36
File access attribute, N4-15. *See also* File open attribute
File access privileges. *See also* Field access privileges
  access levels, N3-26 – N3-27
  cancelling, N3-31
  changing, N3-31
  creating, N3-2 – N3-27
  precedence over field privileges, N3-28
  privilege scheme, N3-21
  restricting, N3-28
  storing, N3-32
  types, N3-5, N3-26
File extensions, L9-16 (table), U1-4 (table), U5-2. *See also* individual extensions under . (period)
File open attributes
  default, N4-3
  definition, N4-2

exclusive, N4-2
by file type (table), N4-4 – N4-5
shared, N4-2
File server, N1-3
Config.sys, NB-3
installing dBASE ADMINISTRATOR, NB-6, NC-5, ND-6
requirements, N1-7 – N1-8
uninstalling dBASE ADMINISTRATOR, NB-15, NC-13, ND-15
FILE() function, P13-4 – P13-5, R4-4, U6-33
Filenames, U1-4, U5-2
Files Menu, PROTECT, N3-22 – N3-30
file privilege scheme, N3-31
field access level, N3-27
field access privileges, N3-27 – N3-29
summary of entries (table), N3-24
Filter conditions, UB-3 (table)
Filter file, *see* Query (.qry) files
Filtering
commands, P10-16 – P10-20
a database file, L5-1 – L5-17. *See also* CREATE/MODIFY QUERY, Query (.qry) file, SET FILTER
in different work areas, P11-11
effect on dBASE commands, UB-3
in views, P11-16
input, P7-1, P9-1
options, establishing, P11-15 – P11-16
FIND, P10-10, U2-32, U5-124 – U5-125
end-of-file, P10-13 – P10-14
isolating next record, P10-12
with memory variables, P10-11
Finding. *See also* FIND, LOCATE, Searching, SEEK
files, P13-4
records, P10-8, L4-1 – L4-11, U2-30 – U2-32
string within string, *see* AT()
FKLABEL() function, P4-6, U6-34. *See also* FKMAX()
FKMAX() function, P4-6, U6-35. *See also* FKLABEL()
FLAG command (Novell network), NC-8
Flag options, R2-3, R3-3
FLOCK() function, N4-10, N5-32 – N5-34
Flow charts, P1-21
Footers in printed output, P12-10
FOR condition, L9-21, U2-3
Form feed command, P12-10, U5-120
Format, date, U5-200. *See also* CTOD(), DTOC()
Format (.fmt) files, P8-12 – P8-17, U1-6, U2-43 – U2-47. *See also* CREATE/ MODIFY SCREEN, Screen form, SET FORMAT
closing, P8-14, U2-35 (table), U2-47, U5-48, U5-53, U5-232 – U5-233
creating, L9-16 – L9-17, P8-12, U2-45 – U2-46, U5-82 – U5-87
with memo fields, P8-16
multiple page, P8-15, U5-20
opening, L9-17, P8-14, U2-34 (table), U2-47, U5-82 – U5-87, U5-232 – U5-233. *See also* APPEND, CHANGE, EDIT
Formatted output, P12-3
Formatting of printed page, P12-8
Forms
data entry, *see* APPEND, CHANGE, EDIT, CREATE/MODIFY SCREEN, Screen forms
label, *see* CREATE/MODIFY LABEL, Label (.lbl) files
report, *see* CREATE/MODIFY REPORT, Report (.frm) files
screen, *see*CREATE/MODIFY SCREEN, Screen forms
FOUND() function, P10-16, P14-4, U6-36
Framework II, P1-4,20
FULL field privilege, N3-5, N3-27
Full-screen operations
cursor control keys, UA-1 – UA-3
data entry, U5-26 – U5-27, U5-40 – U5-42, U5-45, U5-118
determining last key, P9-9, U6-65 – U6-67
exiting, *see* READKEY()
with format files, P8-12
indicating field widths, U5-216 – U5-217
Function(s), L10-5 – L10-7, U2-11, U6-1 – U6-2. *See also* individual functions
classes of, U6-4 – U6-6 (table)
commands requiring lock, N5-4
for conversions, *see* Converting






data type produced by, U6-2 – U6-4 (table)
definition, L10-5, U6-1
entering, U6-2
in expressions, U2-4, U2-5
network, N5-27 – N5-37
numeric, P5-5
in program files, L10-5 – L10-7, U3-6
with relations, P11-14
string, P5-5
template, P7-1,4
with @ commands, U5-14 – U5-15
Function keys, U5-224 – U5-225
determining name of, U6-34
determining number of, U6-35
programming, P4-6, U4-6
resetting to defaults, P13-3
Function option
in PICTURE template, P7-4

## G

Garbage in/garbage out, P1-8
GETENV() function, P16-13, U6-37. *See also* OS()
GETS. *See also* Configuration commands
clearing, *see* CLEAR GETS, U5-47, U5-50
with lock functions, N4-9, N5-4
Global variables, *see* Public variables
GO/GOTO, L9-21, P10-8, P10-9, P11-4, U2-32, U5-126
in The Assistant, L4-1 – L4-3
with end-of-file, P10-13
with SET FILTER, P10-17
Graphics
in screen forms, L2-30 – L2-31, P8-7, U5-23, U5-24, U5-87
Group(s), *see* PROTECT
Grouping
database files in catalogs, L7-11 – L7-14, L9-9 – L9-10
records in a report, L6-8 – L6-10, U5-77 – U5-78

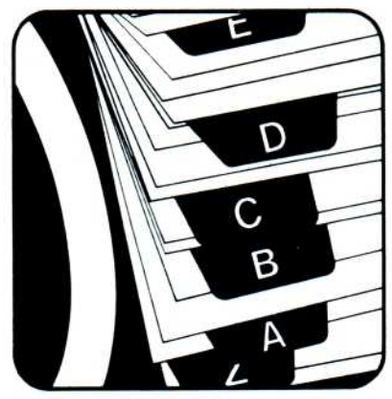
## H

Hardware
requirements, N1-8, N1-9
sharing resources, N1-4
Headers
in database files, P13-7, UC-1
in printed output, P12-10, U5-78
Headings
column, L6-10 – L6-13
field, P4-7, U5-78, U5-178
Help
in The Assistant, L1-15
at dot prompt, L9-3 – L9-4, P4-7
setting up your own, P16-5
HELP, U5-127. *See also* SET HELP
Help.dbs file, P16-16
Hiding a PUBLIC variable, *see* Public variables
Highlight L1-9, U2-12, N3-15
History buffer, L9-6, P15-16, U2-2. *See also* DISPLAY HISTORY, LIST HISTORY
changing size of, L9-6, P15-7, U2-2, U5-237. *See also* SET HISTORY
displaying, U5-107, U5-145
edit commands in, U2-2
executing commands, U2-2
storing command file commands, U5-219
with program testing, P15-7 – P15-8
Horizontal scrolling, P7-6, P8-8

## I

-i option, R2-5, R3-5
IBM PC LAN Program, N1-8
IBM PC network, N1-8, NB-1
IBM PC Network Program, N1-8
Identification functions, U6-6 (table). *See also* individual functions
IDLAN program
definition, N1-6
error messages, NA-1
running on a 3Com network, ND-2
running on an IBM network, NB-1
running on a Novell network, NC-2

IF...ENDIF, P1-14, P2-7, P14-3, P14-4, U3-3, U5-128. *See also* IIF()
IIF() function, P16-1, U6-38 – U6-39. *See also* IF...ENDIF
IMPORT, P16-14, U2-56, U5-124 – U5-125, U5-129 – U5-130
  in The Assistant, L8-2 – L8-4, U5-38
  files created, L8-4
Importing files, *see* IMPORT
Incomplete commands, P15-1
INDEX, L9-22, P10-3, P10-6, P11-4, P11-11, U5-126 – U5-129, U5-131 – U5-134. *See also* SET INDEX, SET ORDER, SORT
  in The Assistant, L4-12 – L4-13, U5-33
  compared with SORT, L4-20
  with FIND and SEEK, P10-10
  on a network, N4-7, N5-4
  with SET RELATION, P11-13
  with ZAP, P11-9
Index (.ndx) file(s), L4-11 – L4-18, U5-131 – U5-134
  active, *see* NDX()
  allowed field types, L4-13
  and file access, N3-5
  changing order, P10-7, U5-247 – U5-248
  closing, P10-7, U2-30, U5-49
  creating in The Assistant, L4-12 – L4-13
  creating at dot prompt, L9-22, U2-28, U5-131 – U5-134
  creating for encrypted (.crp) files, N3-5
  definition, L4-11, U1-6
  determining names, P13-6. *See also* NDX()
  with duplicate records, P10-19. *See also* SET UNIQUE
  index key, U5-2
  key expression, L4-12, U5-131 – U5-134
  to locate records, P10-9
  master (controlling), L4-14, L9-23, P10-7, U5-247
  more than one, L4-13, L4-14 (note) L9-23, P10-6
  multiple-field, P10-6, U5-131
  names in use, P13-6
  opening, L9-22, P10-6, P10-7, U5-239 – U5-240, U5-281 – U5-282
  rebuilding, L4-18 (note), U2-28, U5-172
  with relations, P11-13. *See also* CREATE/MODIFY VIEW, SET RELATION
  selecting, L4-14
  size of, P13-9
  unique, P10-19
  updating, L4-18, L9-23
Index key expression, L4-12. *See also* INDEX, Index (.ndx) files
Indexes, *see* Index (.ndx) files
Indexing a database file, *see* INDEX, Index (.ndx) files, SORT
Information file
  created by dBCODE (.dbg extension), R2-5
  created by dBLINKER (.map extension), R1-9, R3-5
  files, defined, R3-5
  files, referenced, R3-5
INKEY() function, P9-7, P16-11, U6-40 – U6-42. *See also* READKEY()
  values, U6-41 (table)
Input
  accepting, U5-25, U5-135 – U5-136, U5-283
  verifying, P1-6, P9-1, P9-2, P14-3
  determining last key pressed, P9-7, P9-9. *See also* INKEY(), READKEY()
  evaluating special keys, P9-7
  evaluating type, P9-12
  formatting, P7-3
  functions, *see* INKEY(), READKEY()
  pausing for user, P6-16, U5-274
  specifying a range, P7-8
  testing, *see* Testing
INPUT, U5-135 – U5-136. *See also* ACCEPT, WAIT
Input data, *see* Data entry
INSERT, P8-12, U2-23, U5-137 – U5-138
  with BLANK, N5-4, U5-137
Insert mode, L2-10, L9-3
Installation overview, N1-10
Installing dBASE ADMINISTRATOR
  error messages, NA-2
  on a 3Com network, ND-1
  on an IBM network, NB-1
  on a Novell network, NC-1






INT() function, P5-5, P5-18, U6-43. *See also* ROUND()
Integers
converting to, *see* INT()
in memory variables, P3-5
Interrupting program execution, U5-175, U5-274
Inverse video, U5-207, U5-241
ISALPHA() function, P9-12, U6-44
ISCOLOR() function, P4-5, U6-45
ISLOWER() function, P9-12, U6-46
ISUPPER() function, P9-12, U6-47

## J

JOIN, P11-7, U5-139 – U5-142
on a network, N4-7, N5-4
with encrypted files, N5-15
simulation of, P11-14

## K

Key expression, L4-12. *See also* Index (.ndx) files
Key pressed, *see* INKEY(), READKEY()
Keyboard lock indicators, N3-14
Keys. *See also* Function keys
full-screen cursor control keys, U5-156
navigation and editing, U2-2 (table)
MODIFY CMMAND cursor control keys, U5-150

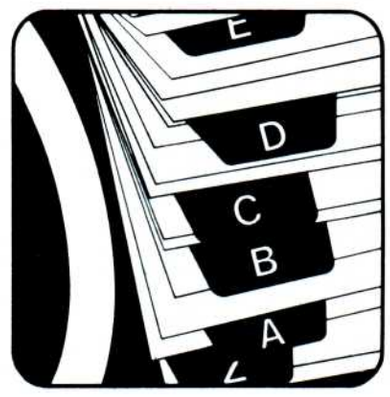
## L

LABEL FORM command, N5-4, U5-143
Label form (.lbl) files, L6-1, L6-28, L9-24, U2-1, U2-43. *See also* CREATE/MODIFY LABEL
commas in, L6-22 – L6-23
contents, L6-21 – L6-24, U5-68 – U5-69
controlling spacing, L6-22 – L6-23
creating in The Assistant, L6-19 – L6-24
creating at the dot prompt, U5-66 – U5-69
displaying, U5-143
modifying, L6-25, U5-66
opening, L6-18
printing, L6-24 – L6-25, U5-137
saving, L6-24
sizes, L6-20 (table), U5-67
LAN, *see* Network
Layout, report, L6-10 – L6-13, U5-68
Leading blanks, P5-11, P5-12. *See also* LTRIM()
Left margin, P12-9
LEFT() function, P5-8, P8-10, P14-7, U6-48. *See also* RIGHT()
LEN() function, P5-6, P8-6, P9-10, U6-49
Length of strings, *see* LEN()
Letter, testing for, *see* ISALPHA(), ISUPPER(), ISLOWER()
Lines, drawing, P8-8, U5-23 – U5-24
Linked file, creating, R3-1 – R3-6
Linking database files, P11-12. *See also* CREATE/MODIFY VIEW, SET RELATION, View (.vue) files
LIST, P5-1, P11-2, P11-16, P15-11, U5-138, U5-144
in The Assistant, L4-9 – L4-11
memo fields, P8-17
with templates, P7-9
List filenames, see Filenames, list
LIST HISTORY, P15-7, U5-145
LIST MEMORY, P1-8, U5-146
LIST STATUS, U5-147
on a network, N2-6, N5-10
LIST STRUCTURE, U5-148
Listing records, *see* Displaying records, Printing records
LOAD, P16-10, U5-149 – U5-152
Local area network (LAN), *see* Network
Local files, N2-5
Local variables, *see* Private variables
LOCATE, L9-21, P10-9, U2-29 – U2-31, U5-153 – U5-154
in The Assistant, L4-3 – L4-7, U5-35
and CONTINUE, L4-6 – L4-7, U2-30 – U2-31, U5-54
comparison with FIND and SEEK, U2-30
with different work areas, P10-10, U5-154
with end-of-file condition, P10-14

Locating. *See also* FIND, LOCATE, SEEK
  next record, U5-54
  specific records, L4-2 – L4-7, U2-30 – U2-32, U5-35, U5-54, U5-153, U5-188
LOCK() function, N4-11, N5-4, N5-36
Locking, N4-10 – N4-14
  in application programs, N4-14
  avoiding deadlock, N4-9
  automatic, commands, N4-7, N5-4
  at the dot prompt, N4-11 - N4-13
  explicit file, N4-8
  files, N4-2, N4-11, N5-32
  functions, N4-11, N5-32, N5-36
  levels, N4-7
  records, N4-8
  related files, N5-32, N5-36
  releasing a lock, N4-12 – N4-13
  shared files, N4-8, N4-17, N5-32, N5-36
  testing for, N4-11
  toggle, N4-12 – N4-13
  values returned, N4-11
LOG() function, P5-5, U6-50. *See also* EXP()
Log in
  Dbsystem.db file, N3-8 – N3-9
  network administrator screens, N3-10
  PROTECTed system, N1-12
  security, N3-2 – N3-4
  unauthorized, N3-10 – N3-11
  user name, N3-3, N3-21, N3-33
Logarithm function, *see* LOG()
Logical field, L1-5, U1-9
Logical operators, U2-6
  in program flow, P1-10, P2-9 – P2-10, P15-3
Logical memory variables, P3-3, U2-10
Login.db file, N1-11
LOGOUT, N4-19, N5-11
Looping, P1-9, P1-11, P2-4
Lotus 1-2-3 files, P16-15, U2-57 – U2-58, U5-28 – U5-31, U5-55 – U5-58
LOWER() function, L4-17, P5-8, U6-51. *See also* ISLOWER(), UPPER()
Lower case. *See also* ISLOWER, LOWER(), UPPER()
  converting from upper case, P5-9, U6-51
  converting to upper case, P5-9, U6-88
  testing for, P9-12, U6-46
LTRIM() function, U6-52. *See also* RTRIM(), TRIM()
LUPDATE() function, P11-7, P13-6, U6-53

## M

M- > (pointer), P3-6, P11-11, U2-9, U2-38
Macro(s)
  in DO WHILE loop, U5-115
  with FIND and SEEK, P10-11
  limitations, R4-3 – R4-4
  in program flow, P3-16
  substitution function, *see* & function
Main program, P1-2, P4-1
MAP command (Novell network), NC-5
MAP SEARCH (Novell network), NC-4
Master (catalog.cat) catalog, U5-197
Matching strings, P10-10
Mathematical functions, U6-5 (table). *See also* individual functions
Mathematical operators, U2-5
MAX() function, P5-5, U6-54. *See also* MIN()
Memo field, U1-9
  adjusting width for output, U5-234
  changing display, P8-17
  editing in The Assistant, L3-11 – L3-12
  editing in programs, P5-16, P6-3, P8-16
  limitation when exporting, L8-5 (note), U5-55
Memo (.dbt) files, U1-6
  backup, *see* Backup files U1-6
  structure, UC-3 – UC-4
Memory, *see* CLEAR MEMORY, DISPLAY MEMORY, LIST MEMORY, RESTORE, SAVE
Memory (.mem) files, P3-11 – P3-15, U1-7, U5-173
  creating, P3-12, U5-187
  not encrypted, N3-5
  for printer configurations, P12-15
  during program development, P3-15
  restoring from, U5-180, U5-181
  saving on disk, U5-187



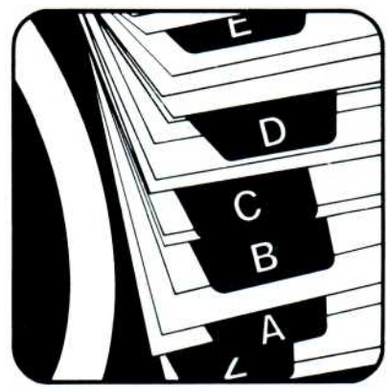



Memory variables (memvars), L9-26 – L9-30, P3-1, U1-10, U2-8 – U2-9
with ACCEPT, P6-13 – P6-16, U5-25
activating, from a memory file, P3-12, U5-180 – U5-181
clearing, P3-14, P13-3, U5-48, U5-51
commands, U5-8
to control program flow, P3-16 – P3-17
creating, L9-27, P3-2
current status, U5-108, U5-146
deleting, U1-10, U5-173 – U5-174
displaying values, L9-27
distinguishing from fields, L9-28 (note), P3-6
editing contents of, U2-9
with FIND and SEEK, P10-11, U5-124, U5-188
hiding PUBLIC, U3-11, P16-5
initializing, L9-27, P3-2, U2-8, U5-262
initializing dates, P5-25, P3-4, U6-19
limitations, P3-6
name length, L9-27
number in memory, L9-27
pointer to, P11-11
PRIVATE, *see* Private variables
in program files, P1-8, P3-12
in program flow, P3-7
PUBLIC, *see* Public variables
releasing, P3-10 – P3-15, U5-48, U5-51, U5-173 – U5-174
restoring, P3-12, U5-180 – U5-181
reusing, P14-7
same as field name, U2-9
saving in a file, L9-29, P3-12, U5-187
for screen displays, P8-5, P8-9, P14-7
size in memory, L9-27, P3-1
specifications, U1-2, U1-10, P3-1, P3-6
types, L9-27, P3-3 – P3-6, U2-9 – U2-10
for updating records, P11-4
with @...GET, P6-9, U5-16
with @...SAY, P8-5, U5-16
Memvars, *see* Memory variables
Menu bar, N3-12, U2-13, L1-8
turning off, P4-8
Menu(s), L1-6, U2-10 – U2-11
abandoning work, U2-19
action line, U2-14
cancelling selections, L1-14
commands which display, 2-12
dBASE program, P1-14
definition, L1-6
dialogue box, U2-14
enumerated values, U2-17
exiting, U2-19
highlight, U2-13
main menu, P1-14, U2-13
menu bar, L1-8, N3-12, U2-13
message line, N3-14, U2-14
navigation keys, U2-13 – U2-14, U5-235
navigation line, N3-14, U2-14
opening, L1-8 – L1-9
option types, U2-16 – U2-19
pull-down, L1-8
saving work, N3-18, U2-19
selecting options, L1-9 – L1-12, U2-16
selection bar, N3-15
status bar, U2-14
structure, U2-13 – U2-14
submenu, L1-9 – L1-12, U2-14
submenu lists, U2-18
user-defined values, U2-18
Merging data, *see* JOIN
Message(s), L1-14, L1-13 (figure), N3-15, P4-9, U2-14. *See also* Prompt, SET MESSAGE TO, SET SCOREBOARD TO
error, *see* Error messages
MESSAGE() function, P15-13, P16-9, U6-55. *See also* ERROR()
Message line *see* Message(s)
on a network, N4-16, N5-35
MIN() function, P5-5, U6-56. *See also* MAX()
MOD() function, P5-5, U6-57 – U6-58
MODE command (DOS), P12-2, U5-252
MODIFY COMMAND, L10-1, P1-3, P15-8, P15-13, U2-44, U2-46, U3-7, U3-8, U5-149 – U5-151, U5-155 – U5-157
with ASCII codes, P8-9
with batch files, P16-15, P16-16
with format files, P8-12, U2-44

program files, P16-3, U3-7, U3-8, R4-3
with a word processor, P1-5, U4-6
MODIFY <filetype>, *see* specific file type
MODIFY LABEL, *see* CREATE/MODIFY LABEL
MODIFY QUERY, *see* CREATE/MODIFY QUERY
MODIFY REPORT, *see* CREATE/MODIFY REPORT
MODIFY SCREEN, *see* CREATE/MODIFY SCREEN
MODIFY STRUCTURE, *see* CREATE <newfile> /MODIFY STRUCTURE
MODIFY VIEW, *see* CREATE/MODIFY VIEW
Modular programming, P1-2, P1-19 – P1-24
in debugging, P15-5
Module, *see* Modular programming
Modulus, determining, *see* MOD()
Monitor
changing color display on, U2-20, U5-204 – U5-208
monochrome, U2-20
set attributes, P4-5, U5-194 – U5-197
set reverse video, P8-2, U5-207, U5-241
special effects on, *see* CHR()
testing for color, *see* ISCOLOR()
Month
from a date, P5-21
determining, *see* CMONTH(), MONTH()
MONTH() function, P5-21, U6-59. *See also* CMONTH()
Multiplan files, P16-15, U2-55, U5-28 – U5-31, U5-55 – U5-58
Multiple choices
conditions, P1-13, P2-10
in program flow, P2-10
Multiple file server networks, N1-3, N2-4
Multiple page forms, P8-15
Multiple page screens, P6-13
MVARSIZ, *see*Configuration commands

## N

Navigation keys, U2-2 U5-156(table)
Navigation line, L1-13 (figure), L1-14, N3-14, U2-14
NDX() function, P13-6, U6-60, U5-247, U5-248. *See also* DBF()
Nesting
expressions in query files, L5-10 – L5-11
in programs, P2-12 – P2-15, U3-3
NET SHARE command (IBM network), NB-9
NET USE command (IBM network), NB-10
Network
administration, N1-1
administration password, N3-9
application planning, N2-4 – N2-5
and dBASE III PLUS, N1-3 – N1-5
commands, *see* dBASE network commands
description, N1-2 – N1-3
driver, N1-3
environment, N1-2
error messages, NA-1 – NA-4
functions, N5-26 – N5-37
hardware/software requirements, N1-7, N1-8, N1-9
installation, NB-1, NC-1, ND-1
log-in, N1-10 – N-11
node, N1-3
programming for, applications, N4-1, N5-3 (table)
programs, N1-5
protection against collision, N4-1 – N4-2
shell, N1-4
software requirements, N1-8
Network administrator
log-in, N3-9
password, N3-9
Networks
3COM 3 + Network, ND-1
IBM PC, NB-1
installation overview, N1-10
Novell, NC-1
New page, *see* EJECT
NONE field privilege, N3-5
NOTE/*/&&, P1-22, U3-4, U5-159






Novell Advanced NetWare/86, N1-8, NC-1
Novell network, N1-8, NC-1
Null string, P9-10
Number(s)
  converting to strings, P5-15. *See also* CHR(), STR()
  maximum, *see* MAX()
  minimum, *see* MIN()
  rounding, *see* ROUND(), INT()
  square root of, *see* SQRT()
Numeric
  accuracy, U1-2
  data, formatting, *see* TRANSFORM()
  memory variables, P3-5, U2-10
  output, display fixed decimals, U5-231
  output, set decimals for, U5-213
  ranges, problems with, P2-9
Numeric fields, U1-9
  calculating totals in, U2-58, U5-276
  finding averages, U5-39
  summarizing data in, L6-25 – L6-27
  totalling, in reports, L6-13, U5-79
  totalling to another database, U5-276

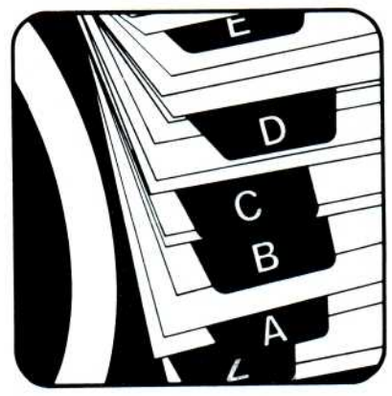
## O

-o option, R2-5
ON, U5-160 – U5-162
  in debugging, P15-11, P15-13
  in linked files, R4-2
ON ERROR, N4-16, N5-30, P9-12, R4-2, U5-161
ON ESCAPE, P9-12, R4-2, U5-160
  **Esc** key ignored, R4-2, U5-160
Opening files, L9-13 – L9-14, U2-33 – U2-34
  in work areas, U2-37, U5-189 – U5-190, U5-281
  in The Assistant, L3-2 – L3-3
  format files, U5-20
  maximum number, U2-33, U5-189
  table of commands, U2-34
Opening menus, L1-8
Operating system. *See also* OS()
  determining, P16-14
  with printers, P12-2
  running from dBASE III PLUS, P16-13, N5-22
  version for networking, N1-7
Operators
  order of precedence, U2-6
  substring, U2-6
  types, U2-5 – U2-6
Options
  dBCODE, R2-1 – R2-6
  dBLINKER, R3-1 – R3-6
Order of arguments, P15-2
Order of precedence, operators, U2-6
Ordering records, *see* INDEX, Index (.ndx ) files, SORT
OS() function, P16-14, U6-61. *See also* GETENV()
Other programs
  running from
    dBASE III PLUS, N2-10, N5-22, P16-12, U5-185 – U5-186
  using data from, *see* IMPORT/EXPORT
OTHERWISE, P2-11
Output, P1-6
  blocks of text, U5-275, U5-278
  left margin setting, U5-242
  device, P12-3, U5-252
  formatted, P12-3
  in program design, P1-19
  to printer, U5-242
  saving on disk, U5-193 – U5-194
  to screen, U5-251
  unformatted, P12-4
Overwrite mode, L2-10, L9-3

## P

-p option, R3-6
PACK, N5-4, P11-8, P11-9, U5-163
  in The Assistant, L3-13 – L3-14, U5-35
Page breaks, P12-16
Page ejects. *See also* EJECT
  suppressing, P12-11
Page format
  in printed output, P12-8
  in reports, U5-72, U5-79

Paper
length, P12-10
position in printer, P12-6
size, P12-6
Parameter(s), L9-3
changing values, P16-8
initializing, P16-7
order, P16-7
passing to named program, P16-6 – P16-9, U5-43, U5-111, U5-164
with procedure files, P16-8
PARAMETERS, P16-7, U5-158, U5-164. *See also* DO
Parsing, P1-3
Password(s), P8-11, P10-3
network administrator, N3-9
user, N3-3, N3-21
Password protection, N3-2, N3-33
PATH command (DOS), P16-13, NB-10, ND-10
Paths, P4-8, P13-5, P13-8. *See also* SET PATH, SET DEFAULT
Pausing for user input, P6-16, U5-283
PCOL() function, P12-16, U6-62
*See also* COL(), PROW() functions
Peripheral devices, N1-3
pfs:File, *see* EXPORT, IMPORT
PICTURE function, U5-18 – U5-19. *See also* @...GET ...SAY, CREATE/MODIFY SCREEN
Pointer, *see* Record pointer
Posting method, P11-7
Precedence of operators, U2-6
Previous versions of dBASE, NB-6, NC-6, ND-6
Print buffer, P12-11
Printer, P12-4
adjust left margin, U5-75 – U5-76, U5-242
column, P12-15
connecting, P12-2
controlling, U2-47
coordinates, P6-6, P12-4, U5-16
direct output to screen and, U5-218, U5-251
ejecting page, P12-10, U5-120
escape codes, P12-12, U2-47, U5-15
head position, P12-11, P12-12. *See also* PCOL()
output, P12-3
output blocks of text to, U5-275, U5-278
parallel, P12-2
ports, N2-9, N5-22, P12-2
route @...SAY and @...GETs to, P12-13, U2-47, U5-16, U5-218
rules for using, P12-2
send ASCII codes to, P12-12 – P12-14. *See also* CHR()
serial, P12-2
special effects, P12-12, P12-13, P12-14
switching, N2-10, N5-22, P12-3
top-of-form setting, P12-6
using several, P12-15
Print head position, P12-11, P12-12
Printing
backwards, P12-5
commands (table), U2-48
customized, U5-15
database information, U5-15, U5-179
elite type, P12-7
envelopes, P12-6
escape code sequences, P12-12, U2-47, U5-16
setting form length, P12-12
format files, U2-43 – U2-45, U5-73 – U5-81
labels, L6-24 – L6-25, U2-43, U5-143
last line, P12-11
null characters, P12-13
pica type, P12-8
program files, PI-5
records, from the dot prompt L9-21
records, from the Assistant, L4-9 – L4-11
reports, L6-16, L9-24, U2-42
responses to commands in programs, U5-261
screen forms, L2-32, U2-47, U5-86
spooled files, N5-20
special effects, P12-12
structure, U5-148
table of commands, U2-49
PRIVATE, P3-8, P3-14, P16-5, U5-159, U5-165
to hide a variable, P16-5






Private variables, P3-8, U3-6
Privilege scheme, N3-4
  creating, N3-4, N3-21 – N3-30
  field access, N3-27 – N3-29
  file level, N3-26 – N3-27
PROCEDURE, P16-3, U5-166
Procedure (.prg) files, P16-3, U1-5, U3-9 – U3-10
  closing, P16-4, U3-11, U5-53
  encrypting, R1-7
  executing, U5-111, U5-166
  for help screens, P16-5
  how to set up, P16-3, U5-166, U5-253
  limitations, P16-4, U5-253
  linking, R1-7, R4-3
  opening, P16-3, U5-245, U5-253
  with PARAMETERS, P16-7
  with RunTime +, R4-3
Profile, user, N3-18 – N3-21
Program, P1-2
  cancel execution of, U5-44
  comments, U5-1, U5-159
  control display of responses, U5-261
  control method of escape, U5-221
  control screen display in, U5-204 – U5-208, U5-218, U5-251
  debugging a, P15-1 – P15-14, U3-9 – U3-10
  designing, P1-17 – P1-24
  echo commands, U5-220
  error recovery, U5-160 – U5-162, U5-183
  execution of, U5-11 – U5-12
  execute a line at a time, U5-260
  locking features, N4-6 – N4-14
  output blocks of text from, U5-275 – 278
  preparing for RunTime +, R1-1, R1-4
  resume execution of, U5-182
  structure, P2-1 – P2-2
  suspend execution of, U5-274
  testing, P1-23, P15-5
Program (.prg) file(s), L10-1. *See also* Command files, Procedure files, MODIFY COMMAND
  advanced features, P16-1, U3-11
  boilerplate, P4-10
  branch to binary (.bin) files, U3-11
  branching execution, U5-111, U5-113, U5-128, U5-154 – U5-156
  closing, U3-8 – U3-9, U5-44
  command file, U3-10
  creating, L10-1 – L10-3, U3-7, U5-155 – U5-157
  debugging, P1-23, P15-1 – P15-14, U3-9 – U3-10
  documentation, P1-23 – P1-24, P15-6
  editing, P1-3, U5-149 – U5-155 – U5-157
  executing, U3-8, U5-112
  functions in, L10-5 – L10-7, U3-6
  long command lines in, L10-7, U5-157
  memvars and, U3-5 – U3-6
  nesting commands, U3-3
  printing, PI-5
  procedure file, U3-10, P16-3 – P16-5
  restore control to calling, U5-184
  running, L10-4, U3-7
  saving to disk, P1-4
  size limitations, P1-5
  structure, U3-4 – U3-5
Programming, P1-2. *See also* Applications Generator
  commands, U3-2 – U3-4, U5-7
  common mistakes, P1-3
  concepts for networks, N4-1 – N4-25
  conditional commands, U5-108, U5-123, U5-154 – U5-146
  control mechanisms, P1-9 – P1-14, P2-3
  controlling loop, P14-2
  data protection, N4-1 – N4-2
  documentation, P1-22
  efficiency in, P16-2
  flow charts, P1-19
  function keys, *see* Function keys,
programming
  housekeeping, P4-12
  main controlling structure, P4-1, P4-11
  main program module, P4-1
  modular, P1-2, P1-19 – P1-24
  network commands, *see* dBASE network commands
  network concepts, N1-3, N4-1
  reusing modules, P1-21, P16-6

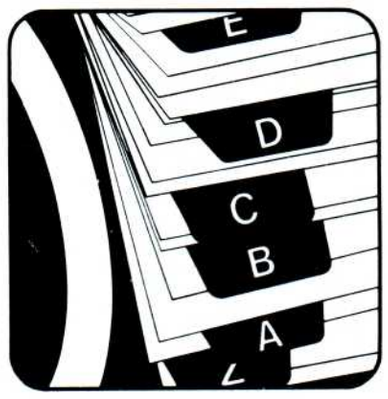
security for network applications, N4-19 – N4-20
toolkits, P1-21
top-down approach, P1-19
working with database files, P10-3
PROMPT, *see* Configuration commands
Prompt
for keyboard entry, U5-25, U5-135 – U5-136, U5-238
setting up for user input, P6-14
PROTECT, N3-1, N3-9, N3-18
abandoning entries, N3-31
access levels, N3-4
assigning file to group, N3-26
definition, N1-5, N2-3, N3-1
entering values, N3-16 – N3-17
Exit Menu, N3-25, N3-31
exiting, N3-31
field access privileges, N3-5, N3-27 – N3-29
fields list, N3-27, N3-29, N3-30
file access privileges, N3-5, N3-24, N3-28
file list, N3-24
Files Menu, N3-22, N3-24
in a non-network environment, N2-2
initiating, N3-9
menu bar, N3-12
menus, N3-11 – N3-18
message line, N3-14
navigation line, N3-14
data encryption, N3-6 – N3-7
password, N3-10
status bar, N3-13 – N3-14
types of security (table), N3-2
user group, N3-3, N3-7, N3-19
Users Menu, N3-18, N3-19 (table)
Protecting data, U2-59 – U2-61, U5-51 – U5-54, U5-55, U5-249
PROW() function, P12-16, U6-63
*See also*PCOL(), ROW() functions
PUBLIC, P3-8 – P3-9, U5-167 – U5-168
Public variables, P3-8, P3-11, P15-2, P16-5
hiding, P16-5
with memory files, P3-14
Pull-down menu, L1-8

## Q

Query clause (?), L9-25, L9-26, U2-54 – U2-56
Query command, *see* ? (query) command
Query (.qry) file(s), L5-1, L9-23 – L9-24, U1-7, U5-70. *See also* CREATE/MODIFY QUERY, SET FILTER TO
creating from dot prompt, U5-70
creating in The Assistant, L5-1 – L5-11, U5-33
entering conditions, L5-2 – L5-9, L5-4 (figure), L9-9 (figure)
modifying, U5-70
nesting expressions, L5-10 – L5-11, U5-71 – U5-72
opening, L5-2, L9-24, U5-32, U5-229
printing, L5-13 – L5-14
storing, L5-13
QUIT, N4-12, P1-6, P2-2, U5-169
Quitting
dBASE III PLUS, L1-26, U5-33
from the dot prompt, L9-30, U5-169
program execution, P1-6, U5-210

## R

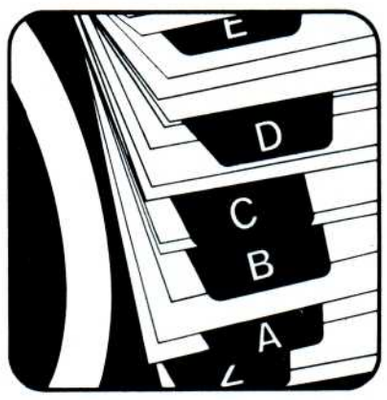
-r option, R2-5, R3-6
RANGE option
to define value limits, U5-17. *See also* @...SAY...GET, P7-8, P14-3
Ranges, U5-17
with dates, P7-8
for limiting input, P9-2
with numeric data, P7-8
READ, P6-9 – P6-13, U5-170
to clear GETS, P6-13, U5-170
in format files, P8-15, U5-16
in multiple-page screens, P16-13
with @...GET, P6-10, P14-3, U5-16
Read file privilege, N3-5, N3-26
READ SAVE, P6-11
Read-only file access attribute, N4-15
Read-only field privilege, N3-5, N3-26
Read-write access attribute, N4-15






READKEY() function, P9-9, U6-64 – U6-66
values (table), U6-65
Rearranging records, *see* INDEX, Index (.ndx) files, SORT
RECALL, P11-8, U5-171
in The Assistant, L3-3 (table), L3-14
on a network, N4-7, N5-4
RECCOUNT() function, P13-7, P14-2, U6-67
*See also* RECSIZE()
RECNO() function, P10-16, U6-68
Record(s), L1-3
adding, in The Assistant, L1-20 – L1-22, L3-7 – L3-9
adding at the dot prompt, P11-5, P11-6, U2-23, U5-26 – U5-27, U5-137 – U5-138
changing, U2-24 – U2-25, U5-40– U5-42, U5-118 – U5-119
copying to other files, P11-10, U5-55 – U5-58
copying from previous, U5-196. *See Also* APPEND FROM; EXPORT; IMPORT
copying fields from, L4-21 – L4-22
counting, U5-62, U6-67
definition, L1-3, L1-4 (figure)
deleting, L3-13 – L3-17, P11-8, U2-25, U5-102, U5-163
displaying, L3-4 – L3-7, U5-4, U5-105. *See also* BROWSE, EDIT
duplicate, P10-19
editing, L3-11 – L3-12, U2-24 – U2-26, U5-41, U5-118, U5-170
filtering deleted, P10-18, U5-215. *See also* SET DELETED
finding, L4-2 – L4-9, P10-8, U2-30 – U2-32
*See Also* FIND; LOCATE; SEEK
grouping, L6-8 – L6-10
in different work areas, P11-11
indexing, see INDEX, Index (.ndx) files, SORT
isolating, P10-8
locating next one, P10-9
merging with other, U2-59, U5-139 – U5-142
moving to, P10-8
number, *see* RECNO()
rearranging, L4-11 – L4-20, U2-26 – U2-30
removing marked, U5-171
retrieving groups of, L4-9 – L4-11
search for, U2-30, U5-124, U5-153, U5-188
size, P13-7. *See also* RECSIZE()
sorting, L4-9 – L4-20, U5-269
total number of, P13-7. *See also* RECCOUNT()
undeleting, U5-171
updating, P11-4, P11-5, P11-7
Record count update interval, *see* SET ODOMETER TO
Record locking, *see* Locking
Record pointer
commands, U5-6
with functions, P11-14
GO/GOTO, U5-126
for memory variables, P11-11
moving in The Assistant, L4-1 – L4-3
in related files, P11-13
with SET RELATION, P11-13
for work areas, P11-11
RECSIZE() function, P13-7, P13-9, U6-69
Redirecting printer output, N2-8, N5-20, U5-252
Re-entering, P15-7, U2-2
REINDEX, N5-4, U2-28, U5-172
Relating database files, *see* View (.vue) files, SET RELATION, CREATE/MODIFY VIEW
Relation chain, L7-6, U5-96 – U5-98
locking, N5-32, N5-36
Relational database files, P10-2
Relational operators, L4-8 (table), U2-5
menu, L3-16 (figure)
Relations, P11-12, U2-38 – U2-41. *See also* SET VIEW TO
closing, P11-13
with functions, P11-14
reusing, P11-15
saving, P11-15, U5-101
setting up, P10-4, P11-15, U5-255 – U5-256
with view files, P11-15, U5-94 – U5-100
Relative addressing, P8-3
with FIND and SEEK, P10-13, P10-14
with printed output, P12-15

RELEASE, P3-10, P3-11, U5-166, U5-173 – U5-174
with ALL, U5-173
ALL with EXCEPT, P3-11
with wildcards, P3-10, U5-173
Releasing
@...GETs, U5-50
fields, U5-49, U5-223
memory variables, 5-48, U5-51, U5-166, U5-173
RELEASE MODULE, P16-10, U5-173
RENAME, P13-10, U5-175
Renaming files, P13-10, U5-34, U5-175
Reordering records, *see* INDEX, Index (.ndx) files, SORT
Repeating characters, *see* REPLICATE
REPLACE, P3-7, P6-2, P11-4 – P11-6, U5-176
in The Assistant, L3-3 (table)
on a network, N4-7, N5-4
Replacing. *See also* REPLACE, UPDATE
fields, P11-4 – P11-6, U5-270
part of string, *see* STUFF(), REPLACE
REPLICATE() function, P8-10, U6-70
REPORT FORM, L9-24, P12-1, U5-171, U5-178 – U5-179
on a network, N4-7, N5-4
Reports, *see* Report form (.frm) files
Report form (.frm) files, U1-7. *See also* CREATE/MODIFY REPORT
entering column headings, L6-10 – L6-11
creating in The Assistant, L6-2 – L6-16
customized, P12-1
definition, L6-1, U1-7
format options, L6-7 (table), U5-75 – U5-76, U5-81 (table)
formatting, L6-6
grouping records, L6-8 – L6-10, U5-77 – U5-78
inserting a column, L6-14
modifying, L6-16, U2-42, U5-74 – U5-81
opening, L6-4
entering page titles, L6-5, U5-75
printing, L6-16, U5-178
saving, L6-16
totalling numeric data, L6-13, U5-79
wrapping text in column, L6-11 (note)
zooming into text entry area, L6-8, U5-74
Requirements
file server, N1-9
network, N1-8
workstation, N1-10
Reserved words, P3-3
Resources, sharing, N1-4
Response file, R3-5, R3-6
creating, R1-3, R1-6
storing, R1-4, R1-6
REST scope, P11-3
RESTORE, P4-10, U5-180 – U5-181
ADDITIVE option, P3-12 – P3-15
RESTORE (DOS), P13-10
RESUME, P15-8, U5-182
RETRY, N4-16, N4-17, N5-12, P16-9, U5-183
RETURN, P2-2, P13-3, P13-4, P14-2, U5-184
with PROCEDURE, P16-3, U3-11
RETURN TO MASTER, P2-4, U5-184
Right margin, P12-8
Right-justifying a string, P8-6
RIGHT() function, P5-8, U6-71. *See also* LEFT()
Ringing the bell, P8-9, U5-195, U5-209
RLOCK() function, N4-11, N5-36 – N5-37
Root file, R3-1, R4-3
ROUND() function, P5-5, U6-72. *See also* INT()
ROW() function, P8-3, U6-73. *See also* COL(), PROW()
RTRIM() function, L10-7, P5-11, U6-74. *See also* LTRIM(), TRIM()
RUN, N2-10, N5-19, U5-185 – U5-186
binary (.bin) files, P16-12
with operating system, N2-10, N5-22
Run time errors, P15-1, P15-3
RunTime +
checkbook management system, R1-4, 7, 10
dBCODE, R2-1 – R2-6
dBLINKER, R3-1 – R3-6
error messages, RA-1 to RA-3
using ***Esc*** key in, R4-2
linkage editor, R3-1
macro limitations, R4-3



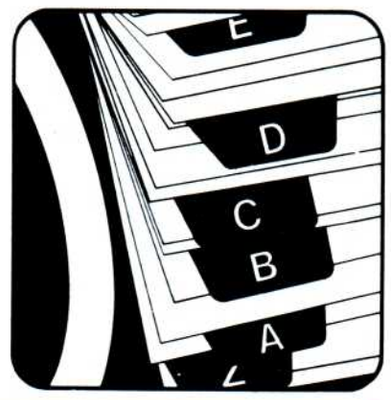



multi-disk applications, R4-4
options, dBCODE, R2-3 – R2-5
options, dBLINKER, R3-3 – R3-6
procedure files, R4-3
program specifications, R4-1
programming tips, R4-6
response file, R1-3, R1-6
root files, R4-3
source code requirements, R4-1 – R4-3
sample session, R1-4 – R1-11
tips, R1-4

## S

-s option, R2-6, R3-6
SAVE, L9-29, P3-12, P12-15, U5-187
Saving
data in The Assistant, L1-25 – L1-26
data entry forms, L2-32
file privilege schemes, N3-18
files, *see* individual file types under . (period)
memory variables, U5-187
output to a text file, U5-193
PROTECT entries, N3-17
report forms, L6-16, U5-79
SET RELATION TO A VIEW, U2-41
user input, *see* ACCEPT, INPUT, WAIT
user profiles, N3-20
Scope, L9-20, U2-3
Scoreboard, P4-9, U5-258. *See also* SET STATUS
with customized screens, P6-5
with status line, P4-9
Scratch files, P11-9, P13-11
Screen
clearing, P6-6, U5-47
controlling, P6-3
coordinates, P6-4, U5-16
direct output to, U5-218, U5-261
graphics characters, P8-7
outputting blocks of text to, U5-275
relative addressing, P8-3
templates, P7-1
top row, P4-9
Screen (.scr) files, U1-7, P8-12
creating, L9-16 – L9-17, U5-82 – U5-87
Screen forms, U2-20 – U2-21. *See also* APPEND, CHANGE, CREATE/MODIFY SCREEN, EDIT
adding a message, L2-18, U5-83 – U5-84, U5-245
adding a title, L2-11 – L2-12
adding fields, L2-3 – L2-6, L2-22 – L2-25, U5-86
advantages, P1-7
centering a title, L2-11
changing field width, L2-19 – L2-21
creating database file in, L2-33
creating at dot prompt, L9-16 – L9-17, U2-43 – U2-47, U5-82 – U5-87
creating in The Assistant, L2-2 – L2-31
deleting fields, L2-25, U5-84
deleting labels, L2-18
deleting lines, L2-11, U5-84
deselecting fields, L2-4 (note)
designing, P7-10
drawing lines and boxes, L2-30 – L2-31, P8-7, P8-8, U5-23 – U5-24, U5-87
editing text, U5-84
increasing field width, L2-19 – L2-21, U5-84
inserting blank lines, L2-10
inserting or deleting spaces, L2-11
labelling fields, L2-17 – L2-19
modifying, *see* CREATE/MODIFY SCREEN
moving fields, L2-12 – L2-17
opening, in The Assistant, L2-2
overlapping fields, L2-15 (note)
PICTURE function, U5-17 – U5-19, U5-86
printing, L2-32, U2-47
in a program file, P6-1 – P6-18, U5-20
prompting for input in, U5-130
RANGE option, U5-86
saving, L2-32
selecting fields, L2-3 – L2-6
stretching or shrinking a box, L2-31 (note)
for updating records, P11-4
Screen Painter, L2-1, P8-12. *See also* CREATE/MODIFY SCREEN
creating a database file with, L2-33

insert mode, L2-10
menus, L2-2 (table)
modifying database file structure, L2-21
overwrite mode, L2-10
PICTURE function option, L2-26 – L2-27, L2-29 (table)
PICTURE template option, L2-28, L2-29 (table)
Range option, L2-28, L2-29 (table), U5-85
Scrolling data entry forms, U5-18 – U5-19
SDF (System Data Format) files, *see* ASCII
Search condition(s)
building a, L4-4 – L4-11
combining, L4-10
definition, L9-21
if not matched, L4-6
specifying, L9-21
Searching. *See also* CONTINUE, FIND, LOCATE, SEEK
in The Assistant, L4-2 – L4-11
and case sensitivity, L4-17
for contents of memory variable, *see* & function
for data, U2-30 – U2-32
an indexed file, L4-14 – L4-17
by matching a search condition, L4-3 – L4-11
for partial string, L4-16
by record number, L4-1 – L4-3
for records, L4-1 – L4-11, P10-8, U2-30 – U2-32, U5-50
for starting position of string within a string, *see* AT()
Security, *see* dBASE security
SEEK, P10-9, P11-4, P14-4, U5-188
in The Assistant, L4-14 – L4-17
with end-of-file, P10-13
with end-of-file condition, P10-14
for indexed files, U2-31 – U2-32
isolating next record, P10-12
with memory variables, P10-11
with partial string, L4-16
SELECT, P10-4, P11-11, P13-5, U2-37 – U2-38, U5-189 – U5-190
Selecting
a database file, L3-2, L9-13, U5-190, U5-281 – U5-282
an index file, L4-13 – L4-14, L9-23, U5-190, U5-239 – U5-240
a format file, L3-3, L9-17, U5-232 – U5-233
Selection bar, N3-15, U2-13
Self-starting programs, *see* Turnkey system
SET, L9-5 – L9-6, P4-3 – P4-9, P13-3, U5-191 – U5-192. *See also* Config.db
commands in Config.db, U4-6 – U4-10 (table)
on a network, N5-13 – N5-14
SET ALTERNATE, P15-12, U5-193 – U5-194
SET BELL, P4-5, P5-14, U5-195
SET CARRY, U5-196
SET CATALOG, L9-9 – L9-10, N4-7, N5-4, U2-52 – U2-55, U5-197 – U5-202
SET CENTURY, P5-22, U5-203
SET COLOR, P4-5, U2-209, U5-204 – U5-208
with passwords, P8-11
SET CONFIRM, P6-13, U5-209
SET CONSOLE, P12-4, U5-210
SET DATE, P5-21, U5-211
SET DEBUG, P15-9, R4-2, U5-212
SET DECIMALS, P5-18, U5-213
SET DEFAULT, P1-6, P4-5, P16-16, U5-214
SET DELETED, P10-18, P13-7, U5-215
SET DELIMITERS, P6-10, P8-1, U5-216 – U5-217
with TO DEFAULT option, P8-2
SET DEVICE, P12-3, P12-4, P12-7, U5-218
SET DOHISTORY, P15-7, P15-10, R4-2, U5-219
SET ECHO, P15-9, R4-2, U5-220
SET ENCRYPTION, N4-21, N5-15 – N5-16
SET ESCAPE, P4-4, P15-11, U5-221
with type-ahead buffer, P8-17
with WAIT, P6-16
SET EXACT, P10-10, P10-18, U5-222
SET EXCLUSIVE, N4-6, N5-18
SET FIELDS, P11-16, U5-223 – U5-228
SET FILTER, P10-17, U5-229 – U5-230, UB-3 (table). *See also* CREATE/MODIFY QUERY, Query (.qry) files



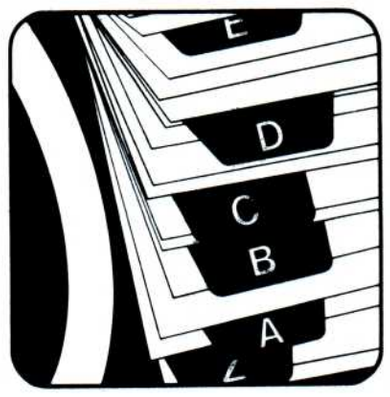



SET FIXED, P5-18, U5-231
SET FORMAT, P8-14, U5-232 – U5-233
SET FUNCTION, P4-6, P13-3, U5-234
SET HEADING, P4-7, U5-235
SET HELP, P4-7, U5-236
SET HISTORY, P15-7, R4-2, U5-237 – U5-238
SET INDEX, P10-7, U2-28, U5-239 – U5-240
SET INTENSITY, P6-10, P8-2, U5-241
SET MARGIN, P12-9, P12-16, U5-242
SET MEMOWIDTH TO, P8-17, U5-243
SET MENU, P4-8, U5-244
SET MESSAGE TO, P4-9, U5-245
SET ODOMETER, U5-246
SET options ignored, R4-1, R4-2
SET ORDER, P10-7, U5-247 – U5-248
SET PATH, P4-8, P13-5, U5-249 – U5-250
  with GETENV() function, P16-14
SET PRINT, P12-4 – P12-5, U5-251
  in debugging, P15-13
SET PRINTER, P12-3, U5-252
  on a network, N2-8 – N2-10, N5-20 – N5-22
SET PROCEDURE, P16-3, P16-4, U3-11, U5-253 – U5-254
SET RELATION, P11-11 – P11-15, U2-38 – U2-41, U5-255 – U5-256
SET SAFETY, P4-8, U5-257
SET SCOREBOARD, P4-9, U5-258
  with customized screens, P6-5
SET STATUS, L9-6, P4-9, P13-3, U5-259
  with customized screens, P6-6
SET STEP, P15-9, R4-2, U5-260
SET TALK, L10-3, L10-4, P4-4, P13-3, U5-261
SET TITLE, U5-262
SET TYPEAHEAD TO, P8-17, U5-263
SET UNIQUE, P10-19, U2-29, U5-264 – U5-265
SET VIEW TO, P11-15, U2-41, U5-266 – U5-267
SHARE command (3Com network), ND-4
Shared directories, NB-9, NC-8, ND-4
Sharing resources
  files, N1-4
  peripherals, N1-4
  software, N1-4
Shell, network, *see* Network shell
Simultaneous data access, N4-2
SKIP, L10-4, U2-32, U5-268
  in The Assistant, L4-3
  backwards, P10-15
  with end-of-file condition, P10-14
SORT, L9-23, P13-6, U2-30, U5-269 – U5-270
  on a network, N4-7, N5-4
Sorted files, *see* SORT, sorting, L4-10
Sorting, U2-30
  in The Assistant, L4-19 – L4-20
  compared to indexing, L4-20
  descending order, L9-23
  records into groups, L6-8 – L6-9. *See also* INDEX, Index (.ndx) files, SORT
Source directory, R1-5, R4-6
Source files
  .src extension for program files, R1-5, R2-2
SPACE() function, P3-4, P5-12, P14-6, U6-75
Space on disk, determining, *see* DISKSPACE()
Spaces
  in command lines, P15-1
  in concatenation, P5-3
  creating blank, *see* SPACE()
Special effects, *see* Printing, special effects
Specifications, U1-1
  command line, U1-2
  database file, U1-1
  field size, U1-1
  file operations, U1-1
  memory variables, U1-2
  numeric accuracy, U1-1
Spooled files, N2-9, N5-20, N5-21
Spreadsheets, *see* APPEND FROM, COPY TO
SQRT() function, P5-5, U6-76
Starting dBASE III PLUS
  with an application program, NB-11, NC-10, ND-11
  with a batch file, NB-12, NC-10, ND-11
  on IBM PC network, NB-10
  on Novell network, NC-9
  in a PROTECTed system, N1-12
  on 3Com network, ND-9
Status bar, L9-2, N3-13, U2-14, U5-259
  in The Assistant, L1-13 – L1-14 (figure)
  controlling display of, L9-6, U5-259

in programs, P4-9
on PROTECT screens, N3-13
STORE, P3-2, P6-2, P11-4, U5-271 – U5-272
STR() function, P5-15, U6-77 – U6-78. *See also* CHR(), VAL()
with decimal places, P5-16
String(s).*See also* Character string, Converting
combining, P5-3, P8-6
changing case, P5-9. *See also* LOWER(), UPPER()
comparing, P5-4, P10-18
String operators, U2-6
Structure
catalog, U2-53, U5-199
command, U2-3
copying, U5-56, U5-57 – U5-58
creating, U5-59, U5-88 – U5-91
database file, U1-8, UC-1 – UC-2. *See also* CREATE < newfile > /MODIFY STRUCTURE
displaying, *see* DISPLAY STRUCTURE, LIST STRUCTURE
memo file, UC-3 – UC-4
Structured programming, P1-21
STUFF() function, P8-6, U6-79 – U6-80. *See also* SUBSTR()
Subdirectories
for network users N2-5
path to, P4-8
with RunTime +, R1-4, R4-6
Subprograms, P1-1, P1-10, P1-17, P2-3
Substring, P5-7
position in string, P5-8
search, P9-3, U6-81
selection functions. *See also* LEFT(), RIGHT(), STUFF()
SUBSTR() function, P5-7, P5-8, U6-81
compared to substring operator, P9-4
Subtracting users
from a 3Com network, ND-15
from an IBM network, NB-14
from a Novell network, NC-13
SUM, N4-7, N5-4, P11-1, U5-273
in The Assistant, L6-25 – L6-27, U5-32
Summarizing data
with COUNT, U5-62
with SUM, U5-273
with TOTAL, U5-276 – U5-277
SUSPEND, U5-274
debugging with, P15-7 – P15-8, P15-10, P15-11
with RunTime +, R4-1
SYLK files, *See also dBASE Bridge*
exporting, U2-57, U5-55 – U5-58
importing, U2-57, U5-28 – U5-31
Symbols
template, P7-1 – P7-4
Symbols and conventions, used in manuals, U5-1
Syntax, command, U2-3
Syntax errors, P15-1, U3-9
System date and time, *see* DATE(), TIME()
System variables, determining, *see* GETENV()

## T

*TechNotes*, P16-17
TEDIT, *see* Configuration commands
Templates, L1-19, P7-1 – P7-7
with character variables, P7-3
for controlling displays, P7-3, U5-19
for converting input, P7-2, U5-19
with date variables, P7-5
functions, P7-4 – P7-5, U5-18 – U5-19
for horizontal scrolling, P7-6
for limiting input, P7-3, P14-3
with literal characters, P7-5
with logical variables, P7-4
with numeric variables, P7-2
symbols, P7-1, U5-16, U5-18 – U5-19
with TRANSFORM() function, P7-9
with ?, P7-8
Testing. *See also* Debugging
for beginning of file, *see* BOF()
for character, *see* ISALPHA(), ISLOWER(), ISUPPER()
for color, *see* ISCOLOR()
for deleted file, *see* DELETED()
for end of file, *see* EOF()






for error, P9-12
for **Esc** key, P9-12
for keypress, *see* INKEY()
for last key pressed, *see* READKEY()
for a lock, N4-11
programs, P1-24, P15-1 – P15-14, R1-15
for result of FIND, LOCATE, SEEK, *see* FOUND()
for ↵ key, P9-10
for **Spacebar**, P9-11
TEXT...ENDTEXT, P6-9, U5-275
Text editor, U3-7, U4-6, U5-155 – U5-157
Text (.txt) files, U1-7. *See also* APPEND FROM, COPY, MODIFY COMMAND, SET ALTERNATE
Time
counting on screen, P9-9
in programs, P5-25
resetting, P16-13
TIME command (DOS), P16-15
TIME() function, P5-24, U6-82. *See also* DATE()
Titles, column, U5-74, U5-197, U5-198, U5-200, U5-262
Toggle locking records, N4-13
Top margin, P12-8
Top-down approach, P1-19, P15-6
Top-of-form, P12-6, P12-11, U5-120
TOTAL, N4-7, N5-4, U2-58, U5-276 – U5-277
with encrypted (.crp) files, N5-15
Trailing blanks
in concatenation, P5-13
removing, P5-10 – P5-12
RTRIM()m U6-74
TRIM(), U6-84 – U6-85
Transferring data between programs, L8-1 – L8-6 *See Also* COPY, EXPORT, IMPORT TRANSFORM() function, P7-9, U6-83
TRIM() function, P5-10, P8-6, U6-84 – U6-85
with ↵ key, P9-10. *See also* LTRIM(), RTRIM()
Turnkey system, P16-15
TYPE command, P1-5, U5-278
Type conversions, P5-2
TYPE() function, P9-12, U6-86 – U6-87
Type-ahead buffer, P8-17. *See also* CLEAR TYPEAHEAD, SET TYPEAHEAD
Typestyle size, P12-8

## U

Unformatted output, P12-4
Uninstalling dBASE ADMINISTRATOR
error messages, NA-2
from a 3Com network, ND-15
from an IBM network, NB-15
from a Novell network, NC-13
Unique index, *see* SET UNIQUE
Unlinking related files, U2-40
UNLOCK, N4-12, N5-24
Unlocking
files, N4-12, N5-32, N5-36
records, N4-12 – N4-13
UPDATE, N4-7, N5-4, P11-7, U5-279 – U5-280
Update file privilege, N3-5, N3-26
Updating
data, P11-4 – P11-8
fields, P11-6
Updating records
adding, L3-7 – L3-10, U5-26 – U5-27, U5-40 – U5-42, U5-137 – U5-138
deleting, L3-13 – L3-16, U2-25, U5-102, U5-163, U5-284
editing, L3-11 – L3-12, U5-118 – U5-119
with fields from another database, U5-279
selecting file for, L3-2 – L3-3
Upper case
converting from lower case, P5-9, U6-88
converting to lower case, P5-9, U6-51
testing for, P9-12, U6-47
UPPER() function, P5-9, P14-5, U6-88. *See also* ISUPPER(), LOWER()
USE, N4-12, P10-3, P11-4, U5-281 – U5-282
USE EXCLUSIVE, N5-25
User
access level, N3-4, N3-20
group, N3-7
group name, N3-7 – N3-81
input, P6-9
interface, P6-1

log-in N3-2 – N3-4, N3-21
profile, N3-4, N3-18 – N3-21
User count
display option, NB-13, NC-12, ND-14
in DBNETCTL.300, N1-7, NB-7, NC-8, ND-8
User-defined values, N3-16, U2-18
User profile(s), N3-4, N3-18 – N3-21

## V

VAL() function, P5-18, U6-89 – U6-90. *See also* STR()
Variables, *see* Memory variables
Verifying input, *see* Input, verifying
VERSION() function, P16-14, U6-91
View (.vue) files, L7-1 – L7-3, U1-8. *See also* CREATE/MODIFY VIEW, SET VIEW TO
appending data in, L7-3 (note)
common field, L7-5 – L7-6, L7-10 (warning), U5-96
controlling file, L7-2, U5-96
creating in The Assistant, L7-3 – L7-9, L5-30
creating at the dot prompt, U2-41, U5-93 – U5-99
creating from the environment, P11-15 – P11-16, U5-101
definition, L7-1 – L7-2, L7-2 (figure)
elements of, U5-94 – U5-95
modifying, U2-41, U5-93
opening, L7-9, L9-25, U5-266 – U5-267
relation chain in, L7-6 (figure), U5-96
saving, L7-9
selecting, L7-9, L9-25, U5-266 – U5-267
selecting data entry form for, L7-8, U5-99
selecting database files for, L7-4 – L7-5, U5-95
selecting fields for, L7-7 – L7-8, U5-97
selectively changing, U5-71 – U5-73
storing a SET RELATION, U2-41, U5-101
VisiCalc files
exporting, U2-56 – U2-57, U5-55 – U5-58
importing, U2-56 – U2-57, U5-28 – U5-31

## W

WAIT, U5-283
changing message, P6-16
Week, day of, *see* CDOW(), DOW()
Wildcards
with DIR, U5-103
with RELEASE, P3-10, U5-173
WKS files
exporting, U2-56 – U2-57, U5-55 – U5-58
importing, U2-56 – U2-57, U5-28 – U5-31
WordStar, P1-4, P16-15
Work areas, P10-4, U2-37
examining, P13-5
with relations, P11-12, U5-101, U5-255 – U5-256
release fields from all, U5-49
resetting to default, P13-3
SELECT, U5-189 – U5-190
select by ALIAS name, U2-38, U5-190
setting up, P11-10
Workstation
Config.sys file, NB-3, NB-4, NC-4, ND-3
definition, N1-3, N2-2 (figure), N2-4 (figure)
requirements, N1-8, N1-9
WP, *see* Configuration commands

## Y

Year, *see* SET CENTURY
from a date, P5-21
YEAR() function, P5-21, U6-92. *See also* DAY(), MONTH()

## Z

ZAP, N5-4, P11-9, U2-25, U5-284
Zoom in/out, *see* CREATE/MODIFY LABEL, CREATE/MODIFY REPORT

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes